ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Laboratorio de Pentesting basado en tecnología de contenedores
(Pentesting lab based on container's technology)

Para acceder al Título de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Miguel Fernández Mata

09 - 2019

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Miguel Fernández Mata
Director del TFG: Alberto Eloy García Gutiérrez
Título: "Laboratorio de Pentesting basado en tecnología de contenedores"
Title: "Pentesting lab based on container's technology"

Presentado a examen el día: 26-09-2019

Para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Miguel Ángel Manzano Ansorena
Secretario (Apellidos, Nombre): José Ángel Irastorza Teja
Vocal (Apellidos, Nombre): Alberto Eloy García Gutiérrez

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

AGRADECIMIENTOS

Después de un largo período de mucho esfuerzo, finalizo este trabajo de fin de grado. Ha sido un tiempo de aprendizaje intenso, no solo a nivel científico, sino también a nivel personal. Este proyecto ha sido muy importante y por eso me gustaría agradecer a todas aquellas personas que me han ayudado y apoyado en este proceso.

En primer lugar, a mi director de TFG, Alberto, sin el cual no habría sido posible, puesto que fue quien tuvo la idea del proyecto, así como el encargado de guiarme durante las numerosas adversidades que surgieron en su desarrollo y las correcciones del mismo.

También quiero agradecer a mi familia, en especial padres y hermana, su apoyo y compresión a lo largo de toda la carrera ha hecho de esta un camino mucho más llevadero. Siempre entendieron el esfuerzo que implicaba y lo tuvieron en cuenta a la hora de prestarme su ayuda.

Así mismo a mi pareja, la cual me ha ofrecido tanto apoyo moral, en momentos duros del proyecto, como logístico, a la hora de darle una mejor apariencia y diseño a ciertos aspectos de este. Sin ella tampoco se habría llegado al resultado actual del trabajo.

Finalmente quiero dar mi agradecimiento a amigos y conocidos que en determinados momentos del TFG me han prestado consejo para un mejor desarrollo de este.

RESUMEN

En la actualidad, las tecnologías de la información y la comunicación (TIC) están sumamente extendidas debido a un crecimiento exponencial que sufrieron en un período de tiempo bastante corto. Esta globalización supuso numerosos beneficios y comodidades en el día a día, tanto de las personas como de las empresas pero, aun así, no todo son ventajas. Dos de los inconvenientes más importantes que surgen a día de hoy son, el consumo de recursos computacionales de forma masiva y el tema de la seguridad. El primero aparece debido a que las aplicaciones y servicios que se utilizan requieren cada vez más cantidad de estos recursos, mientras que, en el caso de la seguridad, los criminales encuentran cómo sacar provecho a estas tecnologías de forma delictiva.

La base de este proyecto se basa en tratar el aspecto del consumo de recursos, planteando los contenedores *Docker* como una alternativa, a corto plazo, a las máquinas virtuales, siendo una opción mucho más óptima que ahorra en memoria, CPU, RAM y ofrece otras muchas ventajas. Así mismo, se aprovecha esta tecnología para abarcar el otro gran problema, la seguridad. Para ello se le da una un enfoque a la utilidad de estos contenedores en el ámbito de la ciberseguridad, proponiendo las auditorías informáticas con esta herramienta como medio para prevenir los ciberataques.

ABSTRACT

Currently, information and communication technologies (ICT) are extremely widespread due to exponential growth that they suffered in a fairly short period of time. This globalization meant numerous benefits and comforts in everyday life; both for people and companies, but even so, not all are advantages. Two of the most important inconveniences that arise today are the consumption of computational resources in a massive way and the issue of security. The first one appears because the applications and services that are used require more and more of these resources, while in the case of security, criminals find how to take advantage of these technologies in a criminal way.

The base of this project is based on treating the aspect of resource consumption, considering Docker containers as an alternative, in the short term, to virtual machines, being a much more optimal option that saves on memory, CPU, RAM and offers many other advantages. In addition, this technology is used to cover the other major problem, security. To this end, an approach is given to the usefulness of these containers in the field of cyber-security, proposing computer audits with this tool as a means to prevent cyber-attacks.

Contenido

1 INTRODUCCIÓN 7
1.1 MOTIVACIÓN Y OBJETIVOS DEL PROYECTO 8
1.2 ORGANIZACIÓN DEL DOCUMENTO 8
2 ASPECTOS TEÓRICOS 10
2.1 SEGURIDAD INFORMÁTICA 10
2.1.1 Desafíos de la Seguridad Informática 11
2.1.2 Posibles Soluciones de la Seguridad Informática 13
2.2 AUDITORÍAS INFORMÁTICAS “PENTESTING” 15
2.2.1 Tipos de Pentesting 17
2.2.2 Métodos de Auditorías Informáticas 18
2.2.3 Fases del Test de Penetración 18
2.3 SISTEMAS VULNERABLES 20
2.4 MÁQUINAS VIRTUALES 21
2.4.1 Tipos de Máquinas Virtuales 21
2.5 CONTENEDORES DOCKER 24
3 ASPECTOS PRÁCTICOS 26
3.1 HERRAMIENTAS DE SEGURIDAD 26
3.2 HERRAMIENTAS DE VIRTUALIZACIÓN 29
3.3 HERRAMIENTAS DE CONTENERIZACIÓN 30
4 DESARROLLO DEL PROYECTO 34
4.1 ESPECIFICACIONES PREVIAS 34
4.2 ESCENARIO DE APLICACIÓN 35
4.3 PENTESTING MEDIANTE MÁQUINAS VIRTUALES 36
4.4 PENTESTING MEDIANTE CONTENEDORES DOCKER 40
4.5 ANÁLISIS DE RECURSOS 44
4.6 PRUEBA DE CONCEPTO DE PENTESTING PARA SU APLICACIÓN EN LOS LABORATORIOS DOCENTES 48
5 CONCLUSIONES Y LÍNEAS FUTURAS 51
6 ANEXOS 54
6.1 ANEXO A 54
6.2 ANEXO B 59
6.3 ANEXO C 66
REFERENCIAS 69

1 INTRODUCCIÓN

Hoy en día las tecnologías de la información y las comunicaciones (TIC) evolucionan a un ritmo vertiginoso. Las personas las hemos convertido en parte esencial de nuestras vidas, hasta el punto de depender completamente de ellas. Todos disponemos de ordenadores en casa, un móvil por persona y multitud de elementos inteligentes capaces de conectarse a internet. Manejamos estas tecnologías con mucha soltura, de manera continuada y, a veces, exponiendo información privada, como datos personales o bancarios, sin ser plenamente conscientes de lo que hacemos. De este desconocimiento es del que se aprovechan los piratas informáticos para poder obtener dicha información mediante numerosos métodos de *hackeo*.

En este proyecto se van a utilizar algunas de las herramientas y métodos usados por los piratas informáticos y profesionales de la seguridad informática para descubrir y prevenir vulnerabilidades del sistema. Aquí, en concreto, las herramientas se emplean para detectar vulnerabilidades en máquinas virtuales, así como en los denominados contenedores. Pese a sus similitudes, ambos mecanismos presentan ventajas e inconvenientes, que deben ser tenidas en cuenta antes de ser aplicados en sistemas en producción. No es de extrañar que las técnicas de ataque y de protección desarrolladas para ser utilizadas sobre máquinas físicas, puedan ser utilizadas ahora sobre este tipo de sistemas. Por lo tanto, desde el punto de vista del usuario, estos dispositivos pueden ser usados como una forma sencilla de entender el comportamiento de los atacantes cuando están enfrente de un sistema real.

Así mismo, tener todos nuestros datos almacenados de manera virtual presenta un problema asociado con el espacio y los recursos que requieren el almacenamiento y procesado de dicha información. Cada vez se hace más necesario almacenar más cosas en los dispositivos, sin borrar lo que ya no se utiliza en la mayoría de los casos. Sin embargo, no solo el espacio es limitado, el resto de recursos (memoria, procesador, etc.) son críticos conforme se actualiza el propio sistema, lo que obliga a alternativas típicas, como son ampliar, optimizar, o incluso, remplazar. Precisamente, basándose en la búsqueda de la optimización, aparecen tecnologías bastante novedosas, prometedoras y, aun, emergentes, como es el caso de la virtualización y los contenedores. Especialmente esta última. Tiene su origen en la empresa *Docker*, que fue la pionera en este ámbito y, por lo tanto, es una de las más asentadas. Esta tecnología revolucionaria se anuncia como una parte crucial en el futuro de la informática y las telecomunicaciones debido a las numerosas ventajas que ofrecen, desde bajos requerimientos por parte de los equipos anfitriones, rapidez de despliegue, gran compatibilidad y capacidad de mantenimiento, simplicidad y configuraciones básicas o gran nivel de seguridad entre otras muchas [1].

El objetivo de los contenedores es implantarse en el ámbito actual de la tecnología como una alternativa viable a las ya asentadas máquinas virtuales. Estas herramientas ofrecen un entorno seguro donde poder probar diferentes métodos informáticos sin afectar al equipo anfitrión, por su parte, los contenedores pretenden hacer lo mismo

de manera más eficiente y cómoda. Aun así, son herramientas aún en desarrollo que tienen una gran progresión para el futuro, así como un gran margen de mejora.

1.1 MOTIVACIÓN Y OBJETIVOS DEL PROYECTO

Vista la previsión de un uso intensivo de los contenedores para la provisión de servicios y aplicaciones avanzadas en Internet, este proyecto hace un estudio acerca de la viabilidad del uso de la tecnología de contenedores *Docker*, para su uso como herramienta para la práctica de actividades de auditoría informática, especialmente para su uso en tareas de aprendizaje, y aprovechar así las ventajas que ofrece esta tecnología frente al uso de máquinas físicas, o la alternativa ya utilizada para este mismo fin, basada en el uso de máquinas virtuales. De hecho, las máquinas virtuales ya aparecen aplicadas en laboratorios docentes, y más concretamente, algunas de estas prácticas incluyen el proceso de auditoría de seguridad.

Con todo lo anterior, el principal objetivo de este trabajo es probar si la tecnología de contenedores es capaz de competir con las ya asentadas máquinas virtuales, especialmente cuando se pretenda realizar las mismas funciones.

Como es lógico, para poder comparar ambas tecnologías, es necesario implementar un escenario de experimentación, que recree las mismas condiciones en cuanto a entradas y resultados del problema que será utilizado como prueba de esfuerzo. Por ello es necesario primero, implementar el escenario de referencia, es decir, basado en máquinas virtuales, y posteriormente el escenario objetivo, es decir, el basado en contenedores. A continuación, se condiciona la prueba de esfuerzo para que se replique alguna tarea relacionada con en el ámbito del hacking ético, para lo cual se sugiere la aplicación de un test de penetración, que será ejecutado en ambos escenarios.

1.2 ORGANIZACIÓN DEL DOCUMENTO

Una vez visto esta primera parte de introducción y objetivos, el capítulo dos da una visión de los principales conceptos teóricos necesarios para entender la base sobre la que se apoya este trabajo, los cuales se dividen en dos grandes temas. En el primero se incluye una definición general de lo que es la seguridad informática la presentación de algunas técnicas de hacking utilizadas hoy en día, hasta llegar a describir la función de las auditorias de seguridad informática como principal solución de seguridad, más concretamente los test de penetración, explicando cómo realizarlos y algunos procedimientos básicos. Así mismo se detallarán los sistemas vulnerables existentes hoy en día para este tipo de análisis. El segundo gran tema, es la definición y descripción de los sistemas de virtualización. Para ello se presentan tanto las máquinas virtuales, como herramientas de simulación, como *Docker*, como plataforma de creación y manipulación de contenedores.

Una vez definido el marco teórico con el que se trabaja, el capítulo tres plantea los aspectos prácticos sobre los que se apoya el desarrollo, definiendo así las herramientas más utilizadas hoy en día en el ámbito de la seguridad, tanto del lado del atacante como por parte de la víctima, en el ámbito de la virtualización y finalmente en el de la *contenerización*. En cada uno de los apartados se plantearán diferentes alternativas, pero siempre explicando el porqué de la elección de uno de ellos, en concreto, para la realización de cada fase del proyecto.

En el cuarto capítulo se expone el desarrollo final del trabajo, explicando cada fase detalladamente, desde los montajes de los laboratorios de *Pentesting* mediante los dos tipos de herramientas previamente mencionadas, hasta los análisis de rendimiento, planteando finalmente un entorno real académico en el que desplegar lo aprendido en el trabajo.

Finalmente, en el último capítulo, se extraen una serie de conclusiones acerca de la experiencia y de los resultados obtenidos, así como los principales puntos abiertos o futuros desarrollos que pueden tomar este trabajo como punto de partida.

2 ASPECTOS TEÓRICOS

En este apartado del documento se tratarán los conceptos teóricos necesarios para la comprensión de este trabajo. Por ello se pone en contexto la situación actual en la que se encuentra la seguridad informática, las principales amenazas a la que se enfrenta, y las posibles soluciones que se podrían tomar, abordando en especial, las auditorías informáticas como la principal solución ante dichos problemas. Además, es necesario incluir la definición de lo que son las máquinas virtuales y para qué sirven, así como de su alternativa, denominadas contenedores.

2.1 SEGURIDAD INFORMÁTICA

La seguridad informática es el área relacionada con la informática y la telemática que se enfoca en la protección de la infraestructura computacional y de todo lo relacionado con esta, especialmente la información contenida en una computadora o aquella que fluye a través de las redes de comunicación. Para ello existe una serie de estándares, protocolos, métodos, reglas, herramientas y leyes concebidas para minimizar los posibles riesgos, tanto a la infraestructura, como a la información [2]. Los ataques informáticos forman parte del mundo de Internet, prácticamente desde la globalización de los ordenadores y de la "red de redes". Personas con inmensa curiosidad por esta ciencia han tratado, desde el principio, de saber más acerca de ella, aunque en ocasiones fuese con fines delictivos. Ante este problema tuvo que surgir gente capaz de contrarrestar los ciberataques, y de esta forma surgió la ciberseguridad. La figura 1 muestra una breve cronología de cómo surgieron los primeros ataques.

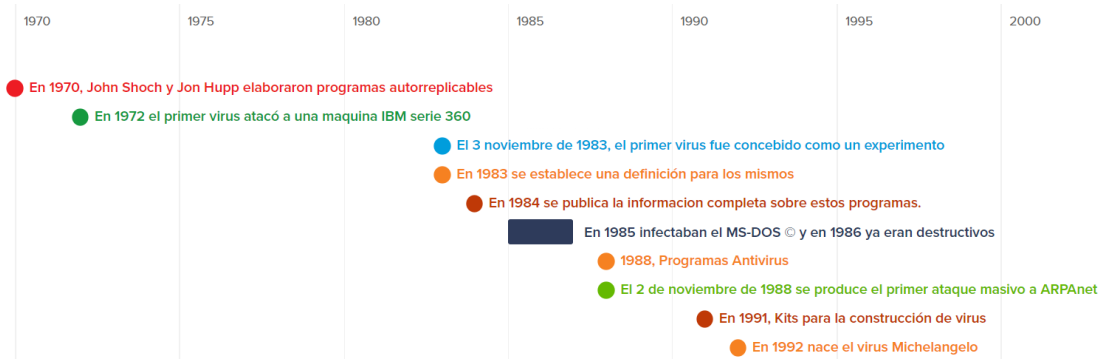

Figura 1: Inicio de los ataques informáticos y su cronología

El cronograma muestra acontecimientos tan importantes como la creación del primer virus y su posterior solución, así como el primer ataque masivo a ARPAnet.

2.1.1 Desafíos de la Seguridad Informática

En la actualidad, la seguridad informática se enfrenta a innumerables métodos de ataque, también denominado **hackeo**, que se aprovechan de cualquier tipo de vulnerabilidad existente en los sistemas para explotar y rentabilizar de muy diferentes formas. Prácticamente, estos ataques se basan en el hecho que todos estamos conectados a la red y manejamos información muy importante con demasiada confianza pero, en muchas ocasiones, no somos conscientes de cuánta información crucial volcamos a la red, así como el nivel de seguridad con el que la transmitimos o guardamos. El problema, hoy en día, radica en que hay disponibles en línea programas, en forma de scripts de código, especializados para realizar actividades de hacking. Son aplicaciones preparadas para que cualquiera las use, incluso principiantes, lo cual significa que casi cualquiera con paciencia, mentalización y motivación puede aprender a *hackear*. Da igual cuál sea la motivación, incluyendo a cualquier persona que quiera acceder a datos personales, credenciales o datos bancarios. Además, al ser código que circulan por la red, cualquier profesional del *hackeo* es capaz de modificarlos para generar otros nuevos, con otros fines, o simplemente, evitar ser descubierto. Algunos de los tipos de ataques más comunes a día de hoy, como se describe en [3], son:

- **Ataques de Phishing**: Es el nombre que reciben los ataques de suplantación de identidad. Son el vector principal de los ataques del denominado malware, y generalmente están compuestos por un archivo adjunto o un enlace incrustado a un correo electrónico, aparentemente inofensivo. Los correos electrónicos de *phishing*, por lo general, afirman falsamente proceder de una empresa establecida o legítima, intentando llamar la atención del receptor. Estos correos generalmente son fáciles de detectar, si sabe qué buscar, ya que, a menudo, presentan errores gramaticales y ortográficos, tienden a solicitar información personal, o invitan a abrir un archivo anexado o un enlace web. Además, generalmente proviene de una fuente que normalmente no requiere esta información, porque ya la tiene porque realmente no lo necesita, o simplemente, no dirige al usuario hacia enlaces externos a través de un correo electrónico. Estadísticamente hablando, a día de hoy, el 91% de los ataques informáticos empiezan por un correo de *phishing*, de los cuales el 30% de estos mensajes son abiertos por los usuarios, y el 12% de ellos acceden al archivo malicioso o enlace adjunto.

- **Ataques de Network-Probes**: Es el nombre que recibe el uso de sondas y analizadores de red. Estos ataques consisten en la colocación de una sonda de red con la cual obtener información suficiente mediante la cual obtener el acceso a una computadora y a sus archivos a través de un punto débil conocido o probable. Las sondas de red no son una amenaza inmediata, básicamente es un monitor de red que analiza protocolos y tráfico de red en tiempo real y, de hecho, son las principales herramientas de diagnóstico y resolución de errores de casi cualquier sistema que utilice una red de comunicación, como Internet. Sin embargo, utilizadas para otros fines, permiten descubrir información sensible en una conexión, bien de los procesos de identificación, o bien de los detalles del hardware y/o software instalado. Cruzando esta información con

bases de datos públicas, es posible identificar puntos de entrada concretos, así como el vector de ataque más adecuado.

- **Ataques de fuerza bruta**: Estos ataques también se conocen como Cracking, y son métodos de prueba y error. Así, por ejemplo, algunos programas de aplicación permiten decodificar datos cifrados, como pueden ser contraseñas o claves de cifrado de datos (DES). Para ello, mediante reintentos exhaustivos, se prueban todas las posibles combinaciones, por ejemplo, de una contraseña, hasta que se pruebe la opción correcta, momento en el que se permite la entrada al sitio que se está atacando. El objetivo de estos ataques, además de contraseñas de acceso o cifrado, también suele ser el descubrimiento de información oculta o no directamente accesible, como es el caso de la estructura de un *Website*, o página Web.

- **Ataques Drive-by download**: Estos ataques fueron realizados originalmente por los denominados Malware, y han sido también nombrados como troyanos. En general, consiste en la descarga automática de un software dañino, a través de la instalación de otro software presuntamente original e inofensivo. En muchos casos, el propio usuario consiente la instalación del uno de ellos, sin saber de la existencia del otro. Actualmente, existe una variante relacionada con el *phising*, ya que simplemente cuando una víctima hace clic en un enlace, involuntariamente y de forma automática, se inyecta el software malicioso en el ordenador u otro dispositivo.

- **Ataques de denegación de servicio (DoS)**: Como su nombre indica, este tipo de ataques consisten en un intento de hacer que un servicio online no esté disponible, ralentizando y/o saturándolo con tráfico de una fuente. Los ataques *DoS* son actualmente uno de los ataques más comunes usados para comprometer los sistemas de una organización, y se basan en el uso de múltiples fuentes, es decir, son ataques *DoS* que usan múltiples sistemas comprometidos para apuntar y someter a un solo sistema. En general, los sistemas comprometidos suelen estar infectados con un troyano, utilizado explícitamente para colapsar un servicio en línea.

- **Ataques de Amenaza Persistente Avanzada (APT):** Bajo este calificativo se agrupan todos aquellos ataques de red en el que una persona no autorizada obtiene acceso a una red y permanece en ella sin ser detectado durante un período prolongado de tiempo. El objetivo de este tipo de ataques es mantener el acceso encubierto y continuo a una red, lo cual permite a los piratas informáticos recopilar continuamente credenciales de usuario válidas y acceder a más y más información valiosa. Al final, estos ataques tienen como objetivo fundamental recopilar información de forma inadvertida, lo que requiere la reescritura continua de códigos y sofisticadas técnicas de evasión.

- **Ataques de ransomware**: Así se denominan a aquellos ataques en los que un software malicioso está especialmente diseñado para bloquear el acceso a un sistema informático, en tanto en cuanto no se pague una suma de dinero. Es

tipo de “secuestro”, se está haciendo popular y los piratas informáticos están reconociendo cada vez más los beneficios financieros de emplear tales tácticas. El ataque basado en *ransomware* ocurre cuando un hacker infecta una pc o servidor, ya sea con un software malicioso cerrando su sistema (*locker-ransomware*) o mediante el cifrado personalizado de archivos importantes en su sistema y exigiendo un rescate (generalmente en *bitcoins*) a cambio de sus sistemas / archivos (*crypto-ransomware*).

Estos son solo algunos de los ataques más comunes hoy en día, pero existen otros muchos tipos de ataques a tener en cuenta, que normalmente implican a varios de los de la lista anterior, y que no hacen más que llamar la atención sobre que toda precaución en la red nunca es poca

2.1.2 Posibles Soluciones de la Seguridad Informática

Lo mismo que Internet evoluciona continuamente, cada día existen nuevos tipos de ataques, así como las correspondientes evoluciones de los ya identificados, y que suelen añadir nuevos niveles de peligrosidad. Sin embargo, y en contrapartida, también existe más gente especializada en saber contrarresta y buscar la forma de detectar estos ataques. Aun así, el número de ataques a día de hoy es abrumador, y por esa razón todas las precauciones que se puedan tomar, ayudarán a evitar un mal mayor.

Tanto si se forma parte de una empresa, como si se es un usuario particular, lo más recomendable es conocer qué tipo de ataques hay, puesto que para cada ataque hay una forma de contrarrestarlo o preverlo. No es de extrañar, que a cada ataque explicado anteriormente, se le pueda asignar una posible solución o defensa, tal como se explica en [3] y se resume a continuación:

- **Ataques de Phishing**: Seguramente cualquier usuario de Internet haya recibido en algún momento alguna de estas recomendaciones:
    - *Comprobar la URL del enlace adjunto en un correo*, pasando el ratón por encima para ver la dirección real y si realmente concuerda con el remitente,
    - *Verificar el remitente* y, en caso de duda de su legitimidad, consultar directamente con él antes de abrirlo.
    - *Nunca enviar información confidencial* por correo, y mucho menos usuarios y contraseñas.
    - *Evitar publicar demasiada información personal* online, puesto que será más fácil elaborar un engaño cuanto más sepan de la víctima.
    - *Usar alguna protección especializada*, en forma de antivirus, antimalware, etc., para que, si por un casual se cae en el engaño, se pueda detectar y detener la infección antes de que cause daños.

  Estas medidas son tan populares que muchas veces acaban siendo ignoradas.

- **Ataques de Network-Probes**: Si bien resulta especialmente difícil, una vez se tenga conocimiento de su existencia, se deberá alertar al equipo de seguridad,

para que pueda realizar un análisis forense y tomar decisiones ejecutivas sobre cómo proceder. Una vez informado, suele ser necesario continuar monitorizando la actividad, mediante el uso de sensores de detección de intrusos adicionales en las secciones de la red descubiertas. La adquisición de detalles sobre el atacante y tratar de determinar qué fue lo que más llamó la atención en un primer lugar, ayudará a prevenir futuras ocurrencias. Esta solución está enfocada a la empresa, puesto que estos ataques en su mayoría son contra compañías.

- **Ataques de fuerza bruta**: Bajo cualquier sistema operativo es necesario activar el modo de autenticación y la configuración de privacidad de bloqueo, ya que son la forma más fácil y efectiva de evitar los intentos de ataque mediante fuerza bruta. Además, es importante diversificar las contraseñas, evitando reutilizar una misma cuenta para servicios diferentes, como es el caso de una cuenta de administrador de dominio que sea reutilizada como una cuenta de conexión de base de datos SQL, ya que esta combinación podría facilitar desde un ataque de fuerza bruta, a una condición de denegación de servicio. Sin embargo, la mejor defensa es la activación del segundo factor de autenticación, abreviado 2FA. Como su nombre indica, es un método para verificar que la persona que está intentando acceder a una cuenta es su verdadero propietario y no alguien que conoce su contraseña, o que la está forzando mediante fuerza bruta. De esta forma, si alguien conoce la contraseña, no podrá acceder a dicha cuenta si no facilita el segundo factor de autenticación que, por ejemplo, suele ser un código numérico temporal, que se genera de forma aleatoria y tiene validez de, normalmente, 30 segundos.

- **Ataques Drive-by download**: La máxima en estos casos es mantener el software lo más actualizado posible e instalar software de filtrado de web o proxys para evitar sitios inseguros. Tradicionalmente, desactivar Java y cualquier ejecución de scripts es otra recomendación típica, pero contrasta con la realidad de los sitios Web, que incluyen gran cantidad de contenidos basados en dichas piezas de código. Por último, es necesario evitar utilizar una cuenta de administrador para uso habitual, ya que, en caso de infección, otorga acceso de administrador a cualquiera que haya provocado el daño.

- **Ataques de denegación de servicio (DoS)**: La manera más fácil, aunque costosa de defenderse, es asegurar el ancho de banda, o en general, la capacidad. Los ataques DoS modernos se están volviendo increíblemente grandes, y con frecuencia sobrepasan con creces las capacidades de los sistemas atacados. Cada elemento de la red es importante, y así, una de las piezas más importantes es el servidor DNS. Su protección y correcto balanceo es la primera medida a tomar, al igual que el filtrado de protocolos no permitidos y mensajes basura en los routers. Básicamente, es necesario bloquear todo lo que se pueda, y deba, directamente en el borde de red.

- **Ataques de Amenaza Persistente Avanzada (APT)**: Las empresas que son víctimas de los ataques APT normalmente se centran en el nivel empresarial y

son propensas a los riesgos cibernéticos, debido a los productos o servicios políticos, culturales, religiosos o ideológicos. Dado su gran impacto y complejidad, los ataques APT requieren hacer uso de múltiples tecnologías para combatirlos. Se necesitan equipos de seguridad específicos, con monitorización las 24 horas del día, el uso de cifrado de datos extremo a extremo, la segregación de la red, los sistemas de detección de intrusos y las listas blancas de aplicaciones. Como medidas preventivas, tener un "sistema de gestión de vulnerabilidades implementado, mantener los parches de seguridad actualizados y probar continuamente el estado de la seguridad en las infraestructuras de TI, son algunas de las más utilizadas.

- **Ataques de Ransomware**: Al igual que en el caso del *phising*, el uso del conocimiento y la concienciación para evitar los correos electrónicos falsos y deshabilitar las extensiones de archivos ocultos son la mejor forma de defenderse contra el *ransomware*. Sin embargo, la única solución confiable es la de realizar y mantener copias de seguridad de todos los archivos, que permitan restaurar todo el sistema minimizando al máximo las pérdidas.

Pese a todo, todo lo anterior no es más que una serie de recomendaciones y buenas prácticas, a tener en cuenta para una mayor seguridad en la red. Como en todo, siempre hay una forma más eficiente de medir la seguridad de una infraestructura, como es el caso de las denominadas auditorías informáticas. En concreto, las relacionadas con la seguridad de la red, son llamadas test de penetración, y desde el punto de vista de la empresa, siempre será la mejor solución para saber cuan segura es la red, al permitir encontrar los puntos débiles y así corregirlos.

2.2 AUDITORÍAS INFORMÁTICAS “PENTESTING”

Hoy en día todo el mundo conectado se enfrenta a riesgos, cada vez más frecuentes en la red, que pueden afectar a los sistemas informáticos. Ser consciente de estos riesgos es fundamental y, por eso, cada vez más empresas recurren a las auditorías de seguridad informática, mediante las cuales analizar sus infraestructuras de red. Estas auditorías son realmente test de penetración, también denominados “*Pentesting*”, que consisten en atacar un sistema informático, de mutuo acuerdo con el cliente, para identificar sus fallos, vulnerabilidades y demás errores de seguridad existentes, para así poder prevenir futuros ataques externos.

Los importantes ataques y filtraciones sufridos por varias empresas en los últimos años, han hecho que las auditorías de seguridad, pese a ser algunos tipos muy novedosos, estén actualmente en auge. Han proliferado herramientas que permiten comprender la gravedad de los peligros a los que una organización se enfrenta en el día a día. Un golpe de clic permite detectar brechas de seguridad pre-existentes en una determinada compañía, y así prever los riesgos que pueden surgir.

La mayoría de los test de penetración son realmente procesos completos de hacking, solo que, realizados bajo un amparo legal, al contar con el consentimiento de los

propietarios de los equipos que se van a testear. Además, en todos los casos se evita provocar cualquier daño real sobre los sistemas analizados. En definitiva, se simula el comportamiento de un intruso con el fin de poder adelantarse a sus siguientes movimientos. Así, la mayoría de estos test están diseñados para clasificar y determinar el alcance y la repercusión de los fallos de seguridad. De esta forma, se obtiene conocimiento y consciencia del peligro que existe en el sistema testeado, de las defensas de las que dispone y de su eficiencia. Además, también proporcionan datos de probabilidad de éxito ante un ataque al sistema, así como de otras vulnerabilidades existentes, que no podrían ser descubiertas de ninguna otra forma [4].

Si bien el término *pentesting* resulta cada vez más conocido, hay toda una terminología especializada alrededor de este campo, tal como refleja [5], y de los cuales es necesario destacar:

- **Vulnerabilidad**: Es cualquier fallo en la seguridad de una aplicación, sistema o hardware, y que más coloquialmente es conocido como agujero, precisamente porque es el lugar (o forma) por donde colarse en un dispositivo o sistema para hacerse con el control de una aplicación o incluso del equipo por completo. Las vulnerabilidades pueden ser desde fallos en la programación de un código, el uso de contraseñas muy débiles (del típico estilo "1234", "password" o "contraseña") o incluso fallos en el diseño de los sistemas físicos (como es el caso de los problemas de desbordamiento de un buffer de información del sistema).

- **Exploit**: En general es cualquier medio que permite explotar una determinada vulnerabilidad. Originalmente el término *exploit* se reservaba a las vulnerabilidades asociadas con el sistema operativo, por lo que eran críticas al no poder ser evitadas sin actualizaciones específicas de todo el sistema. Sin embargo, con la proliferación de sistemas propietarios, y sobre todo la globalización de los lenguajes de programación, actualmente se denomina *exploit* a cualquier aplicación, programada con el fin único de acceder a un sistema que contenga una determinada vulnerabilidad, adquirir su control y/o provocar un funcionamiento indebido. Tal es su popularidad, que existe una herramienta especializada en el desarrollo y aplicación de *exploits*, denominada *Metasploit*. Parte de un proyecto *Open Source* que originalmente recopilaba vulnerabilidades e informaba de éstas. Posteriormente, colaborando con las compañías afectadas, se procedió al desarrollo de mejoras en los sistemas de detección de intrusos y *malware*. Con el tiempo, la librería de *exploits* ha crecido en base a tres tipos fundamentales. En función de dónde se ejecute éste:

    - ***Exploit Remoto***: Se puede ejecutar desde una red interna o bien desde Internet para poder acceder al sistema víctima.
    - ***Exploit Local***: Para ejecutar este tipo de *exploit*, es necesario tener acceso al sistema vulnerable, previamente a su ejecución, aunque también puede ejecutarse tras acceder a la máquina haciendo uso de un *exploit* remoto.

- ***Exploit del lado del cliente***: Es el tipo de *exploit* más usado, puesto que aprovecha vulnerabilidades existentes en aplicaciones que se encuentran instaladas en la mayoría de equipos de usuarios finales. Suelen llegar al equipo mediante correos electrónicos, pendrives o mediante una "navegación insegura".

- **Payload**: Es un término al que no se puede dejar de hacer mención siempre que se habla de *exploits*. Por definirlo de una forma simple, un *payload* es una pequeña aplicación que aprovecha una vulnerabilidad afectada por un *exploit* para obtener el control del sistema víctima. Lo más común en un ataque es aprovechar una vulnerabilidad con un *exploit* básico, para inyectar un *payload* con el que obtener el control del equipo al que se ataca. De hecho, un subproyecto de *Metasploit* es el principal referente en este ámbito, el denominado *Meterpreter*. Con esta solución, se puede cargar *payloads* que permiten realizar multitud de acciones sobre la víctima, desde acceder al sistema de archivos del equipo, a incluso poder ver lo que muestra la pantalla del ordenador atacado.

A modo de resumen, el *pentesting* va a llevar a cabo un estudio de las vulnerabilidades de un determinado sistema mediante la comprobación del mayor número de *exploits*, incluyendo todos los *payload* relacionados, de entre la librería de elementos conocidos.

2.2.1 Tipos de Pentesting

Según el grado de conocimiento que se tenga acerca del sistema al que se vaya a testear, hay diferentes formas de realizar estas pruebas, tal y como se plantean en [4], y que son:

- **Pentesting de caja blanca**: en este tipo de test se conoce todo acerca del sistema, la aplicación o la arquitectura testeada. Es el análisis más completo, ya que parte de un análisis integral, que evalúa toda la infraestructura de red. Al disponer de un volumen alto de información, suele realizarse directamente por miembros del propio equipo de TIC de la empresa.

- **Pentesting de caja negra**: en este caso, sin embargo, no se dispone de ningún tipo de información sobre el objetivo. Es una prueba a ciegas, por lo que es lo más cercano a lo que podría realizar un atacante externo. Bien realizado, es el método más completo, ya que se asemeja a la forma de actuar de los cibercriminales, aunque requiere de muchísimos recursos si se quiere hacer bien.

- **Pentesting de caja gris**: este estudio es una mezcla de los dos anteriores, es decir, se parte de cierta información privilegiada, mediante la cual poder identificar vulnerabilidades y amenazas reales, reduciendo el tiempo y los recursos necesarios para llevar a cabo un análisis y obtener un informe de actuación no solo preventiva, sino incluso reactiva.

La elección de uno u otro tipo solo depende de las condiciones determinadas en base a lo que establezca el cliente.

2.2.2 Métodos de Auditorías Informáticas

Una vez se ha determinado el tipo de prueba a realizar, de las tres indicadas anteriormente, el siguiente paso es elegir el método a seguir. Esta elección se basa en las necesidades existentes y en función de las características del sistema, o de los requerimientos de la empresa. Estos son algunos de los métodos de auditorías de seguridad informática según [4]:

- **ISSAF (Information Systems Security Assessment Framework)**: organiza la información alrededor de los criterios de evaluación, escritos y revisados por expertos.

- **PCI DSS (Payment Card Industry Data Security Standard**): este método fue desarrollado por las compañías de tarjetas de débito y de crédito más importantes y sirve como guía para las organizaciones que procesan, almacenan y transmiten datos de los titulares de las tarjetas.

- **PTES (Penetration Testing Execution Standard)**: es puesto en práctica por muchos profesionales altamente reconocidos del sector, además de ser un modelo a seguir en libros de aprendizaje asociados al *pentesting*.

- **OSSTMM (Manual de la Metodología Abierta de Testeo de Seguridad)**: sus pruebas no son especialmente innovadoras, pero es uno de los primeros acercamientos a una estructura global de concepto de seguridad. Este modelo es referente en instituciones que precisan de un *pentesting* de calidad, ordenado y eficiente.

Estos métodos son extensos y densos de tratar, pero conocerlos en profundidad es una necesidad a la hora de aplicarlos. En este trabajo se ha hecho uso del método PTES, al ser el más relacionado con los procesos de *pentesting*.

2.2.3 Fases del Test de Penetración

Un test de penetración se compone de múltiples etapas, en la que se realizan diferentes tipos de actividades en distintos ámbitos y entornos. La profundidad con que se lleven a cabo las actividades dependerá de ciertos factores, entre los que se destaca el riesgo que puede generar hacia el cliente alguno de los métodos que se aplican durante la evaluación. Como refleja [6], las principales fases a seguir son:

- **Contacto**: En esta fase inicial se debe acordar con el cliente en qué va a consistir el test de penetración, entendiendo cuál es el objetivo de este *pentest*, cuales son los servicios críticos para la empresa y qué supondría un mayor problema en caso de ataque. En esta fase se ha de hablar y acordar por

escrito diversos aspectos del test de penetración, así como un escrito por parte del cliente que autorice el test y limite la responsabilidad en caso de que surjan problemas.

- **Fase de recolección de información**: Esta fase del *pentest* se dedica a obtener toda la información posible de la empresa, disponible a través de información pública de Internet, arañas (*crawlers*) y escáneres (*probes*), mediante la cual hacerse una idea de los sistemas y programas en funcionamiento. El origen de la información puede ser muy variopinta, ya que, por ejemplo, la propia actividad de los empleados de la empresa en redes sociales puede revelar muchos detalles que no serían visibles de otra manera.

- **Fase de modelado de amenazas**: En este momento, y a partir de la información recogida previamente, se debe pensar, como si de un atacante se tratase, cuál va a ser la estrategia de penetración, cuáles deben ser los objetivos y de qué manera se puede llegar hasta ellos.

- **Fase de Análisis de vulnerabilidades**: Llegados a este punto se debe valorar el posible éxito de las estrategias de penetración elegidas, a través de la identificación proactiva de vulnerabilidades. Ahora es cuando la habilidad del *pentester* se pone de manifiesto, ya que la creatividad del mismo es determinante para seleccionar y utilizar correctamente todo el arsenal de herramientas a su disposición, para conseguir los objetivos establecidos en pasos anteriores.

- **Fase de Explotación**: Una vez detectadas vulnerabilidades viables, es el momento de intentar conseguir acceso a los sistemas objetivo del test de penetración. Para ello se ejecutan *exploits* o, simplemente, se utilizan credenciales robadas (obtenidas por cualquier medio) para ganar acceso a los sistemas.

- **Fase de Post-Explotación**: Una vez se haya accedido a los sistemas del cliente, comienza la fase en la que se debe demostrar qué podría suponer esta brecha de seguridad para el cliente. En esta fase se trata de conseguir el máximo nivel de privilegios, información de la red y acceso al mayor número posible de sistemas, identificando qué datos y/o servicios hay al alcance.

- **Fase de Informe**: Como punto final, hay que presentar el resultado de la auditoría al cliente, de manera que este comprenda la seriedad de los riesgos emanantes de las vulnerabilidades descubiertas. Es fundamental remarcar aquellos puntos en los que la seguridad se había implantado de manera correcta y aquellos que deben ser corregidos, así como de qué manera. Esta fase es para las dos partes posiblemente la más importante.

Si bien este sería el esquema lógico de actuación de un *pentest*, es decisión, consensuada entre ambas partes, del *pentester* focalizar el estudio a una o varias de

las fases anteriores, en función de las razones por las cuales se haya decidido realizar dicha prueba de estrés de los sistemas de seguridad de la empresa.

2.3 SISTEMAS VULNERABLES

En seguridad informática, la palabra vulnerabilidad hace referencia a una debilidad en un sistema permitiendo a un atacante violar la confidencialidad, integridad, disponibilidad, control de acceso y consistencia del sistema o de sus datos y aplicaciones. Las vulnerabilidades son el resultado de errores o fallos en el diseño del sistema, aunque, también pueden ser el resultado de las propias limitaciones tecnológicas, ya que, en principio, no existe sistema 100% seguro [7].

El objetivo de las auditorias de seguridad es el análisis y gestión de sistemas llevado a cabo por profesionales para identificar, enumerar y posteriormente describir las diversas vulnerabilidades que pudieran presentarse en una revisión exhaustiva de las estaciones de trabajo, redes de comunicaciones o servidores. Como se ha mencionado previamente, su principal objetivo son los equipos, las redes y los servidores, es decir, todos aquellos objetivos de los que se puede extraer un beneficio para el atacante, o provocar un gran perjuicio a la víctima.

Durante muchos años, los objetivos se focalizaban en ordenadores físicos conectados a través de Internet. Sin embargo, la tecnología evoluciona y, ya no existen únicamente dispositivos en forma de ordenadores hay dispositivos portátiles, equipos industriales, sistemas domóticos o inclusos sensores que se conectan a la red, y por lo tanto son potenciales víctimas para ser atacados. Pero ya no es solo eso, todos estos dispositivos hacen referencia a equipos físicos, y han dado paso a los dispositivos virtuales. Estos, trabajan de forma *virtualizada*, en los denominados entornos *cloud*, que permiten tener todos los recursos, principalmente el almacenaje, de forma remota en un servidor externo, conectados a través de internet.

Las empresas dedicadas a la provisión de plataformas de virtualización en la nube son plenamente conscientes de lo insegura que puede ser esta nueva configuración de los sistemas en red, y es por ello que son las primeras que recurren a auditorías de seguridad informática que les permitan ser conscientes de las vulnerabilidades que puedan tener sus servidores y cómo contrarrestarlas. Según [8], existen dos problemas de seguridad críticos al mover los servicios o las cargas de trabajo en un entorno cloud, el primero es la forma de acceder de forma segura a los servicios, mientras que el segundo desafío, y el más importante, es cómo extender las políticas de seguridad de la empresa a la nube. Debido a estos problemas, surgen los cuatro pilares sobre los que se apoya la seguridad en cualquier entorno cloud:

- **Defensa del host**: tanto si los servicios se están ejecutando en las instalaciones o en la nube, una organización necesita para reforzar la máquina virtual (VM) mediante el uso de protección basada en el equipo, como, por ejemplo, antivirus, *antispyware* y sistema de prevención de intrusiones en el host. Esto

puede ser complementado además con filtrado de contenidos web, y capacidades de monitorización de los registros.

- **Visibilidad y control de accesos**: Se debe proporcionar una capa adicional de protección para asegurar el resto de la infraestructura de red y de las cargas de trabajo, mediante la administración de todas las cuentas de usuario y los correspondientes servicios de autenticación y autorización.

- **Encriptación**: llevada a todos los contenidos y servicios, permite asegurar los activos de la empresa donde quiera que estén.

- **Simplificación operativa**: la seguridad tiene que ser coherente, transparente, y operativamente sencilla de gestionar, tanto si las cargas de trabajo e están ejecutando en el centro de datos, en una nube privada o en una infraestructura de nube pública.

En cualquier caso la seguridad llega hasta los dispositivos virtuales entre los que actualmente se encuentran las máquinas virtuales y los contenedores, y que son motivo de análisis en este trabajo,.

2.4 MÁQUINAS VIRTUALES

Una máquina virtual es el resultado de ejecutar un software que crea una capa independiente donde se emula el funcionamiento de un ordenador real con todos los componentes de hardware que necesita para funcionar (disco duro, memoria RAM, tarjetas de red, tarjeta gráfica, etc.). Este software puede ser ejecutado en casi cualquier sistema operativo o programa, y la máquina resultante se comporta tal y como lo haría un ordenador real. Toda esta emulación se encapsula en una serie de archivos, desde donde se ejecuta la máquina virtual, en forma de proceso con incluso su propia ventana gráfica, como si de un programa más se tratara. La gran ventaja es que nada de lo que suceda en el interior de esa ventana (realmente en el proceso) tendrá por qué afectar al equipo que la ejecuta. La única diferencia entre la máquina virtual y el equipo físico donde se ejecuta, es que el ordenador en sí posee hardware real, mientras que la máquina virtual emula todos sus componentes de forma que no tiene por qué corresponderse con el hardware físico que tiene instalado el anfitrión. Aunque estas herramientas emulan a un ordenador físico, también requieren de parte de los recursos físicos del anfitrión, esto es, cada máquina virtual consumirá cierta cantidad del procesador, de memoria RAM, espacio en el disco duro, y parte del procesamiento de la tarjeta gráfica que permita soportar la interfaz de la máquina virtual [9].

2.4.1 Tipos de Máquinas Virtuales

Las máquinas virtuales, según se indica en [10], se pueden clasificar en dos grandes categorías, según su funcionalidad y su grado de equivalencia a una verdadera máquina:

- **Máquina virtual de sistema**: También llamadas máquinas virtuales de hardware, permiten a la máquina física subyacente multiplicarse entre varias máquinas virtuales, cada una ejecutando su propio sistema operativo. A la capa de software que permite la virtualización se le llama monitor de máquina virtual o *hipervisor*. Un monitor de máquina virtual puede ejecutarse bien directamente sobre el hardware, o bien sobre un sistema operativo. Un ejemplo gráfico se mostraría en la figura 2.

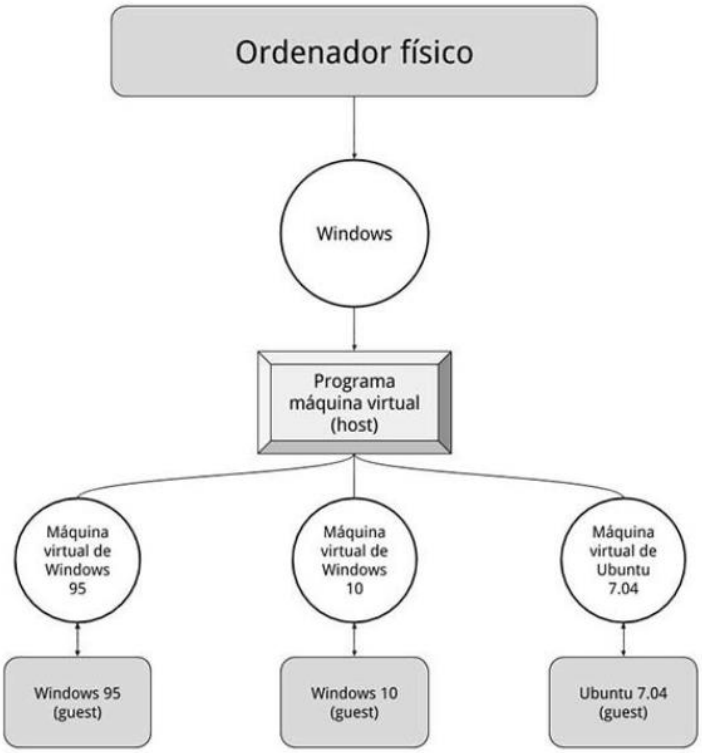

Figura 2: Esquema de la máquina virtual de sistema.

Como se puede observar en el esquema, el ordenador anfitrión, con su propio sistema operativo, da soporte a multitud de máquinas virtuales, tantas como el hardware del anfitrión permita, cada una de ellas con un sistema operativo independiente, pudiendo ser completamente diferentes según los intereses del usuario.

- **Máquina virtual de proceso**: También llamada "máquina virtual de aplicación", se ejecuta como un proceso normal dentro de un sistema operativo, sirviendo de enlace entre un lenguaje de programación y el propio sistema, realizando una interpretación u otra técnica de enlace entre fuente y código máquina. La máquina se inicia automáticamente cuando se lanza el proceso que se desea ejecutar, o manualmente para ejecutar código interactivamente. Por otra parte, se detiene cuando el subproceso es finalizado o se le pide terminar al entorno de ejecución. Su objetivo es el de proporcionar un entorno de ejecución independiente tanto de la plataforma de hardware, como del sistema

operativo. De esta forma, los detalles de la plataforma subyacente permanecen ocultos y permite que un programa genérico se ejecute siempre de la misma forma sobre cualquier plataforma. El ejemplo más conocido actualmente, de este tipo de máquina virtual, es la máquina virtual de Java, que interpreta un código intermedio entre Java y código máquina. El diagrama del ejemplo se muestra en la figura 3.

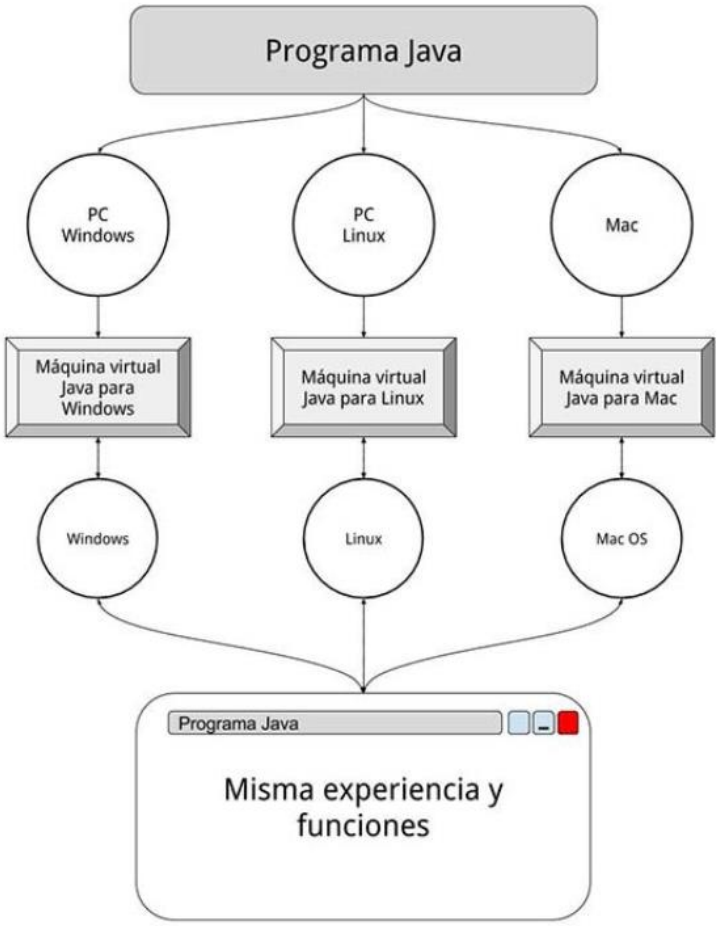

Figura 3: Esquema de la máquina virtual de proceso

Como se muestra en el esquema, un mismo programa, que es ejecutado en distintos sistemas operativos, puede ser virtualizado, de tal forma que se ejecute de la misma forma en todos los equipos, independientemente del sistema operativo del anfitrión.

En cualquier caso, la creación de máquinas virtuales que recrean sistemas operativos es algo costoso a nivel de recursos, memoria RAM, capacidad de procesamiento o almacenamiento, puesto que el sistema operativo virtualizado se ejecuta sobre el sistema operativo host, y tendrá su propio *kernel* y un conjunto de bibliotecas y dependencias. Los recursos que utiliza cada máquina virtual son finitos, pero son recursos que se restan de la capacidad total de la máquina que las soporta. Por ello, no es una solución muy eficiente en algunos contextos [12]. Una idea de su arquitectura se refleja a continuación, en la figura 4.

Figura 4: Esquema de un ejemplo de la arquitectura de las VMs

El *hipervisor* se encarga de establecer el entorno de ejecución adecuado para cada máquina virtual, actuando como intermediario entre los sistemas operativos virtualizados, y el sistema operativo y el hardware del anfitrión. Cada máquina virtual hace uso de un entorno de procesamiento y ejecución totalmente independiente del resto, incluido el del anfitrión.

2.5 CONTENEDORES DOCKER

En la actualidad, tener una máquina virtual implica una necesidad de hardware de altas prestaciones o un despliegue relativamente amplio para montar dichos recursos. No es de extrañar que, con el desarrollo de los sistemas de virtualización, también aparecieran alternativas que redujeran dichos requerimientos. Este es el caso de las tecnologías de contenerización, de entre las cuales destaca *Docker*.

Docker es una plataforma de virtualización, a nivel de sistema operativo, que permite crear una aplicación y empaquetarla junto con sus dependencias y librerías en forma de "contenedor". Este contenedor, de puro software, a su vez puede ejecutarse en otras máquinas, sin más que intercambiar el mencionado contenedor. En realidad, la *contenerización* es la evolución natural del concepto de virtualización, introduciendo importantes diferencias que la hacen especialmente indicada para determinadas aplicaciones. En esencia, permite realizar las mismas tareas de cualquier máquina virtual, pero está especialmente indicado para la virtualización ágil de procesos [11].

En el caso de los contenedores, el hecho de que no necesiten un sistema operativo completo, sino que reutilicen el subyacente, reduce mucho la carga que debe soportar la máquina física, el espacio de almacenamiento utilizado y el tiempo necesario para lanzar las aplicaciones. Por lo tanto, los contenedores son mucho más ligeros que las máquinas virtuales. Si cuando se define una máquina virtual es necesario indicar de antemano cuántos recursos físicos se le van a dedicar. En el caso de los contenedores

esto no es así, de hecho, no se especifican qué recursos se van a necesitar, sino que es el propio gestor, denominado *Docker Engine*, el equivalente al *hipervisor* de las máquinas virtuales, el que, en función de las necesidades de cada momento, es el encargado de asignar los recursos necesarios para que cada contenedor funcione adecuadamente. Además, cada contenedor incluye las librerías específicas para cada aplicación, sin que sean compartidas con otros contenedores, es decir, son unidades autocontenidas. Todo esto hace que los entornos de ejecución de *Docker* sean mucho más ligeros, y que se aproveche mucho mejor el hardware. Así, por ejemplo, una misma máquina física permitirá mantener muchos más contenedores que VMs, y mientras que una VM puede tardar un minuto o más en arrancar y tener disponible una determinada aplicación, un contenedor *Docker* se levanta y responde en unos pocos segundos [12].

La arquitectura de *contenerización* puede resumirse en la estructura de la figura 5.

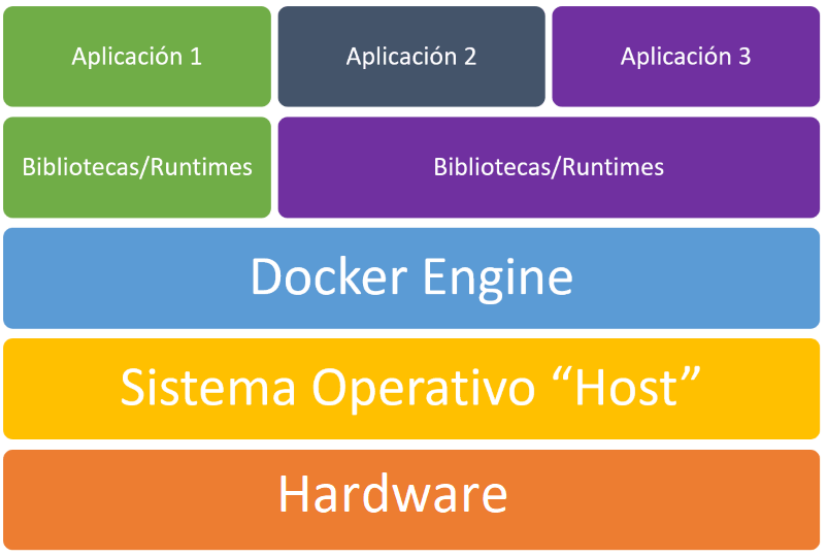

Figura 5: Esquema de un ejemplo de la arquitectura de los contenedores

El sistema de gestión de contenedores actúa como intermediario entre las aplicaciones finales y el sistema operativo y hardware del anfitrión. Cada contenedor presenta, al igual que en las máquinas virtuales, un entorno de ejecución independiente del resto, por lo que se mantiene un aislamiento entre sistemas virtualizados especialmente interesante desde el punto de vista de la seguridad.

3 ASPECTOS PRÁCTICOS

En este apartado se explican de una forma más práctica aquellos componentes de la seguridad informática que han sido tomados como referencia en este trabajo. Se destacan las herramientas y entornos más utilizados a día de hoy por los profesionales de la seguridad. De la misma forma, es momento de introducir los programas que se utilizan en la actualidad para tareas de virtualización y *contenerización*.

3.1 HERRAMIENTAS DE SEGURIDAD

La mejor herramienta para evitar este tipo de incidentes es ser cauto en la red y no fiarse de nada de lo que se tenga la más mínima sospecha. Aun así, existen algunas herramientas que pueden ayudar a proteger o a analizar las vulnerabilidades de los equipos conectados, es decir, permiten *hackear* los equipos como si de un atacante se tratase, y así averiguar qué debilidades existen. A continuación, y de acuerdo con [13], se relacionan los 10 sistemas operativos más utilizados para la realización de *hacking* ético y *pentesting* a día de hoy:

- **Kali Linux**: Desarrollado por *Offensive Security* y sucesor de *BackTrack*, la distribución Kali Linux encabeza la lista de los mejores sistemas operativos para fines de hacking. Este sistema operativo basado en *Debian* viene con más de 600 herramientas de *Pentesting* preinstaladas, que la hacen ser la más amplia caja de herramientas de seguridad. Estas herramientas, muy versátiles, se actualizan periódicamente y están disponibles para diferentes plataformas como *ARM* y *VMware*. Para un trabajo forense, este sistema operativo de hacking viene con una función de arranque en vivo, que ofrece un entorno perfecto para la detección de vulnerabilidades. Kali Linux está basado en un modelo *rolling reléase*, es decir, un sistema de software en constante desarrollo, el cual es instalado una primera vez y posteriormente es actualizado sobre el mismo, en vez de tener que reinstalarlo sucesivamente. El acceso a su web oficial es [14].

- **Parrot Security OS:** Basado en *Debian*, ha sido desarrollado por el equipo de *Frozenbox*. Esta distribución de Linux está diseñada para hacking ético, pruebas de penetración, informática forense y criptografía, entre otros. En comparación con otros, el sistema operativo *Parrot Security* ofrece un sistema operativo ligero que es altamente eficiente. Junto con su abundancia de herramientas legalmente reconocidas, también aporta la posibilidad de trabajar y navegar de forma anónima. Realmente, *Parrot Security OS* es una mezcla de *Frozenbox OS* y *Kali Linux*, y de hecho, utiliza los repositorios de Kali para la actualización de sus herramientas. Viene con entorno de escritorio MATE e interfaz de gran alcance que se deriva de la famosa *Gnome 2*. Este sistema operativo para *pentesting* es altamente personalizable y también con una gran comunidad de soporte. El acceso a su web oficial es [15].

- **BackBox Linux**: es un sistema operativo basado en Ubuntu con enfoque en evaluación de la seguridad y pruebas de penetración. *Backbox Linux* viene con una amplia gama de herramientas de análisis de seguridad que ayudan en el análisis de aplicaciones web, análisis de redes, etc. Es una distribución muy apreciada por de muchos hackers y viene con un entorno de escritorio completo. Los repositorios de software de las herramientas de *hacking* incluidas se actualizan regularmente con las versiones más estables. El acceso a su web oficial es [16].

- **Samurai Web Testing Framework**: es básicamente un entorno Linux Live que viene preconfigurado para funcionar como plataforma de *Pentesting web*. El Framework contiene varias herramientas de hacking libres y de código abierto para la detección de vulnerabilidades en sitios web. A menudo es considerado como el mejor sistema operativo para pruebas de penetración web. El acceso a su web oficial es [17].

- **Pentoo Linux**: Basada en *Gentoo Linux*, *Pentoo* es un sistema operativo de pruebas de penetración que está disponible como un Live CD instalable de 32 y 64 bits. Esta distribución, basada en XFCE, viene con soporte de persistencia, que permite guardar todos los cambios realizados antes de extraer la memoria USB. Este excelente sistema operativo para hacking viene con una amplia variedad de herramientas ordenadas en categorías como *exploit*, cracker, base de datos, entre otras. Al ser un derivado de *Gentoo* hereda el conjunto de características del mismo e incluye una configuración adicional. El acceso a su web oficial es [18].

- **DEFT Linux**: La distribución *open source* de Linux DEFT significa *Digital Evidence and Forensic Toolkit*. Está basado en Ubuntu y construido alrededor del software DART (*Digital Advanced Response Toolkit*). Viene con muchas herramientas forenses populares y documentos que pueden ser utilizados por hackers éticos, expertos en pruebas de penetración, especialistas en seguridad IT y otras personas que se dedican a actividades similares. El acceso a su web oficial es [19].

- **Caine**: es una distribución centrada en la seguridad, basada en Ubuntu, que está disponible como *Live CD*. Es sinónimo de *Computer Aided Investigation Environment* y también se puede ejecutar desde un disco duro después de su instalación. Esta distribución de Linux viene con una amplia gama de herramientas para ayudar en la investigación forense de sistemas. *Caine* viene con un gran número de aplicaciones de bases de datos, memoria, forenses y de análisis de red, aunque también cuenta con aplicaciones más comunes como navegadores web, clientes de correo electrónico y editores de documentos para los propósitos de computación habituales. El acceso a su web oficial es [20].

- **Network Security Toolkit (NST)**: es una distribución *Linux* basada en *Fedora*, que se ejecuta en plataformas de 32 y 64 bits. Este *Live CD* fue creado para dar acceso a las mejores aplicaciones de seguridad de red de código abierto para propósitos de pruebas de penetración. Esta distribución de hacking, fácil de usar, convierte cualquier sistema x86 en una máquina de hacking ético que puede ayudar en la detección de intrusos, monitorizar el tráfico de red y escanear redes y equipos. El acceso a su web oficial es [21].

- **BlackArch Linux**: está disponible como una distribución completa de Linux para los investigadores de seguridad y hackers éticos. Se deriva de *Arch Linux*. El constante crecimiento de este sistema operativo, útil para fines de hacking, se complementa con más de 1,400 herramientas, que han sido meticulosamente probadas antes de ser añadidas. El acceso a su web oficial es [22].

- **Bugtraq**: Disponible en *Debian*, *Ubuntu* y *openSUSE*, *Bugtraq* es famoso por su lista de correo electrónico, que se dedica exclusivamente a la seguridad informática. Los temas que se tratan incluyen discusiones sobre vulnerabilidades, los anuncios relacionados con la seguridad, métodos de explotación, etc. El equipo *Bugtraq* consta de hackers experimentados y desarrolladores que ofrecen un gran servicio para hackers éticos y *pentesters*. *Bugtraq* viene con varias herramientas de p*entesting*, incluyendo herramientas móviles forenses, herramientas de prueba de software malicioso y otros programas desarrollados por la comunidad *Bugtraq*. El acceso a su web oficial es [23].

Aunque cualquiera de los sistemas listados anteriormente podría ser utilizado sin problemas, para la realización de este trabajo se ha elegido **Kali Linux**, primero por ser el más completo a día de hoy, y segundo, por haberse utilizado durante el curso académico, por lo que es un entorno ya conocido y más amigable.

La otra cara de la moneda la constituyen aquellos sistemas operativos que, por diferentes razones, presentan más vulnerabilidades de lo normal. Suelen ser utilizados como objetivos durante el aprendizaje o testeo de otras herramientas, lo que hace más sencillos completar estos análisis. Algunos de los sistemas vulnerables más comunes, según [24], son:

- **Penterlab**: Esta suite, contiene hasta 15 máquinas virtuales que incluyen las vulnerabilidades más comunes de cada sistema diferente. Los problemas no son emulados, ya que se utilizan sistemas reales con vulnerabilidades reales. Cada máquina virtual contiene un documento pdf explicando cómo explotar la vulnerabilidad en caso de no encontrar solución. El acceso a su web oficial es [25].

- **Metasploitable**: máquina virtual vulnerable basada en *Ubuntu* que fue creada por el equipo de *Metasploit* para resolver los problemas de aprendizaje en el marco de *Metasploit*. Se centra en las vulnerabilidades de la capa de red, ya

que contiene todos los servicios vulnerables para que se pueda probar la penetración. El acceso a su web oficial es [26].

- **Hackxor**: es realmente una aplicación web que contiene secuencias de comandos que son vulnerables a *Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF*), estructurado *Query Language Injection* (SQLi), Inyección de comandos remotos (ICE), y muchos más. Esta máquina se ejecuta en *Fedora* 14. El acceso a su web oficial es [27].

- **Kioptrix**: una máquina que tiene tres imágenes diferentes y ofrece desafíos que exigen al atacante tener un acceso de *root* usando cualquier técnica. El acceso a su web oficial es [28].

- **NETinVM**: es una imagen *VirtualBox* o *VMware* que contiene una serie de máquinas virtuales *Kernel Virtual Machine* (KVM) que, cuando se inician, conforman una red informática completa dentro de la máquina virtual. Esta red ha sido concebida principalmente como una herramienta educativa para enseñar y aprender sobre sistemas operativos, redes de computadoras y seguridad de sistemas y redes. El acceso a su web oficial es [29].

- **UltimateLAMP**: una imagen VM vulnerable basada en Ubuntu que tiene un servidor LAMP, que se centra en las vulnerabilidades web. Se ejecuta en las versiones más antiguas y vulnerables de *Joomla, Bugzilla, Drupal, phpMyadmin, WordPress, Mutillidae, Moodle* y otros sistemas conocidos de gestión de contenidos. El acceso a su web oficial es [30].

En este caso, el sistema elegido es **Metasploitable**, por ser el más completo y, además, compatible con *Metasploit*. Al igual que en el caso de *Kali*, fue utilizado durante el curso por lo que ya era un entorno conocido.

3.2 HERRAMIENTAS DE VIRTUALIZACIÓN

Una vez conocidas las herramientas de seguridad a utilizar, es necesario seleccionar los sistemas que van a ser utilizados tanto para el ataque, como para la defensa, así como los supuestos objetivos. En este trabajo se analizan los sistemas generados mediante virtualización. Las máquinas virtuales son muy útiles, debido a que permiten tener un entorno a medida dentro de otro PC, manteniéndolo en todo momento controlado y completamente aislado, principalmente, para evitar problemas en el equipo anfitrión. Por esa razón es interesante conocer cuáles son las herramientas que existen actualmente para la virtualización de equipos. Algunos de los entornos de virtualización más utilizados, han sido estudiados en [31], y se presentan a continuación:

- **VMware vSphere Enterprise**: Se trata del producto estrella de una de las principales compañías en el ámbito de la virtualización. Sus sistemas de virtualización sirven tanto para ordenadores de escritorio, como para sistemas

de servidores, y es el software de virtualización más utilizado por las empresas. El acceso a su web oficial es [32].

- **Citrix XenServer**: Citrix, junto con *VMWare*, se coloca en lo más alto, siendo otra de las grandes compañías de virtualización. *Citrix* está basado en software de código abierto y dispone de dos versiones: una de pago y otra libre. *Citrix XenServer* es una plataforma de virtualización de servidores administrada, completa e integrada en el potente hipervisor *Xen*. La tecnología *Xen* proporciona aislamiento seguro, control de recursos, garantías de calidad de servicio y migración de máquinas virtuales en caliente. *XenServer* está diseñado para una gestión eficiente de los servidores virtuales de Windows y Linux. El acceso a su web oficial es [33].

- **Microsoft Hyper-V Server**: Es el sistema de virtualización de Microsoft. Una de las funcionalidades que incorpora *Hyper-V Server* es la migración "en vivo". Esta funcionalidad permite mover máquinas virtuales en ejecución desde un servidor físico a otro sin que los usuarios se vean afectados. *Hyper-V* permite crear y administrar un entorno informático virtualizado mediante la tecnología de virtualización integrada en Windows Server. El acceso a su web oficial es [34].

- **VirtualBox**: Es un programa gratuito y de código abierto creado por Oracle. Es una solución bastante recomendada a la hora de virtualizar, ya que incluye casi cualquier entorno: *VirtualBox* puede ser usado en anfitriones Windows y Linux, y puede correr sistemas virtuales *Linux*, *BSD*, Windows e incluso *Mac*. Se caracteriza por ser muy fácil de usar gracias a su función "*Guest Additions*". El acceso a su web oficial es [35].

- **Kernel-based Virtual Machine (KVM**): es un software de virtualización libre y de código abierto para Linux, que se basa en las extensiones de virtualización de hardware Intel *VT-X* y *AMD-V*, y una versión modificada *QEMU*. *KVM* permite ejecutar máquinas virtuales utilizando imágenes de disco que contienen sistemas operativos sin modificar. Cada máquina virtual tiene su propio hardware virtualizado: una tarjeta de red, discos duros, tarjeta gráfica, etc. El acceso a su web oficial es [36].

Si bien cualquiera de los anteriores sistemas podría haber sido utilizado en este trabajo, **VirtualBox**, al ser una herramienta de uso gratuito, muy sencilla e intuitiva, al igual que **Kali** y **Metasploitable**, son usadas durante el curso académico, por lo que no precisan de ninguna preparación previa.

3.3 HERRAMIENTAS DE CONTENERIZACIÓN

Si la virtualización en general resulta importante, en los últimos años ha surgido la **contenerización** como una alternativa que reduce considerablemente los requerimientos de los equipos físicos anfitriones. En el ámbito de los contenedores,

Docker es la opción más nombrada, siendo el principal referente, y sobre el que se apoyan la mayoría de las aplicaciones actuales. A pesar de ello se encuentran algunas aplicaciones alternativas. La mayoría de código abierto y en continuo desarrollo, que ofrecen servicios similares a los de *Docker*. Algunos ejemplos de estas aplicaciones, según [37], son:

- **CoreOS rkt**: es el gestor de contenedores de próxima generación para clústeres de Linux. Diseñado para la seguridad, la simplicidad y la capacidad de composición dentro de las arquitecturas de clúster modernas. *Rkt* descubre, verifica, recupera y ejecuta contenedores de aplicaciones con aislamiento conectable. Además, puede ejecutar el mismo contenedor con diversos grados de protección, desde el espacio de nombres de usuario, a nivel de sistema operativo y el aislamiento de capacidades, hasta la virtualización de hardware más pesado a nivel de máquina virtual. El acceso a su web oficial es [38].

- **Mesos Containerizer**: es un administrador de clústeres que simplifica la complejidad de ejecutar aplicaciones en un grupo de servidores compartidos. *Mesos* proporciona una *contenerización* liviana y el aislamiento de los recursos de los anfitriones utilizando una funcionalidad específica de Linux. Se puede configurar para que los operadores puedan habilitar selectivamente diferentes anfitriones. El acceso a su web oficial es [39].

- **LXC Linux Containers**: es una solución de virtualización ligera basada en el kernel de Linux. Prácticamente se ejecuta sobre el Sistema Operativo, lo que le permite ejecutar varias distribuciones aisladas al mismo tiempo. La diferencia entre la virtualización de *LXC* y *KVM*, es que LXC no emula hardware, sino que comparte el mismo espacio de nombres del *kernel*, como sucede con los contenedores de *Docker*. El acceso a su web oficial es [40].

- **OpenVZ**: es una virtualización basada en contenedores para Linux. *OpenVZ* crea múltiples contenedores de Linux seguros y aislados en un único servidor físico, lo que permite una mejor utilización del servidor, y garantiza que las aplicaciones no entren en conflicto. Cada contenedor se realiza y ejecuta exactamente como un servidor independiente. Un contenedor puede reiniciarse de forma independiente y tener acceso de *root*, usuario, direcciones IP, memoria, procesos, archivos, aplicaciones, bibliotecas del sistema y archivos de configuración. El acceso a su web oficial es [41].

- **Containerd**: Un entorno de ejecución de contenedores estándar de la industria, con énfasis en la simplicidad, la robustez y la portabilidad. *Containerd* admite imágenes OCI, es decir, imágenes provenientes de *Oracle Cloud Infraestructure*, y además está diseñado para funcionar en conjunto con gRPC, el cual es un protocolo de tipo RPC (*Remote Procedure Call*) enfocado a la conexión de microservicios. Así mismo viene con muchas funciones de administración del ciclo de vida del contenedor. El acceso a su web oficial es [42].

Como aplicación derivada del uso de contenedores, el concepto de la "orquestación" de aplicaciones ha tomado un nuevo sentido. La orquestación, actualmente, significa disponer de alguna herramienta o sistema que automatice el despliegue, la gestión, el escalado, la interconexión y la disponibilidad de aplicaciones basadas en contenedores. Algunas de las herramientas que mejor se complementan con *Docker* para gestionar la orquestación de sus contenedores, según [43] son:

- **Kubernetes**: es de hecho el motor de orquestación de contenedores más popular que existe en el mercado. Comenzó siendo un proyecto de Google y miles de equipos de desarrolladores lo usan para desplegar contenedores en producción. *Google* afirma que ejecuta miles de millones de contenedores usando *Kubernetes* cada semana. La herramienta funciona agrupando los contenedores que componen una aplicación en unidades lógicas para una fácil gestión y descubrimiento. El acceso a su web oficial es [44].

- **Docker Swarm**: *Swarm* es la solución que propone *Docker* ante los problemas de los desarrolladores a la hora de orquestar y planificar contenedores a través de determinados servidores. *Swarm* viene incluido junto al motor de *Docker* desde la versión 1.12.0, y ofrece muchas funciones avanzadas integradas, como el descubrimiento de servicios, balanceo de carga, escalado y seguridad. *Swarm* sigue la filosofía de *Docker* de centrarse en la simplicidad y en la experiencia del desarrollador. Se podría decir que es más fácil de usar que *Kubernetes* de inicio, pero no tan potente y no tan adoptado por las empresas, los proveedores Cloud o por la comunidad. El acceso a su web oficial es [45].

- **Mesosphere DC/OS**: El sistema operativo *Mesosphere Datacenter (DC/OS)* es una plataforma de código abierto, integrada para datos y contenedores desarrollados sobre el *kernel* de sistema distribuido *Apache Mesos*. Se ha diseñado para gestionar múltiples máquinas dentro de un centro de datos, en forma de uno o más clústeres, ya sea en la nube o usando software en servidores localmente. DC/OS puede desplegar contenedores y gestionar tanto aplicaciones sin estado como protocolos con estado en el mismo entorno. Es capaz de integrarse con *Docker Swarm y Kubernetes*. El acceso a su web oficial es [46].

- **Azure Container Service (AKS)**: Hace relativamente poco se le cambió el nombre de marca de *ACS* a *AKS*. El servicio de *Azure* es código abierto y está optimizado para su uso en las máquinas virtuales de *Azure*, denominadas *Azure* Virtual Machines. Proporciona las herramientas necesarias para crear, configurar y gestionar la infraestructura abierta de contendedores *Docker*. AKS ofrece desarrollo simplificado de aplicaciones basadas en contenedores, soportando el despliegue para *Kubernetes*, *Mesosphere* DC/OS, o *Swarm*. Escala y orquesta usando las herramientas de gestión de aplicaciones elegidas arbitrariamente y establece la conexión mediante puntos de acceso API estándar. El acceso a su web oficial es [47].

- **Amazon ECS**: El servicio de *AWS* para orquestación de contenedores, *Amazon ECS*, es un sistema de gestión muy escalable que permite a los desarrolladores ejecutar aplicaciones en contenedores sobre instancias EC2. Está formado por muchos componentes integrados que permiten la fácil planificación y despliegue de clústeres, tareas y servicios *Docker*. Aunque, como principal desventaja, no existe soporte para ejecutar contenedores fuera de EC2, los aspectos a favor incluyen ventajas propias del servicio AWS tales como *CloudTrail, CloudWatch, Elastic Load Balancing*, etc. El acceso a su web oficial es [48].

Si bien la orquestación de aplicaciones no ha sido un tema contemplado en este trabajo, el uso de *Docker* para el despliegue de conjuntos de contenedores en forma de prácticas virtuales, es un ejemplo con vistas a futuro para la aplicación de alguna de las herramientas anteriores para, por ejemplo, automatizar el despliegue de todos los componentes necesarios por cada alumno.

4 DESARROLLO DEL PROYECTO

Una vez claras las bases teóricas y el entorno práctico necesario para poder llevar a cabo el proyecto, se procede a la explicación del desarrollo del mismo, comenzando por un repaso a las características del sistema sobre el que se pondrá en funcionamiento el laboratorio de *pentesting* sobre el que se han realizado todas las pruebas. Para ello se parte de un primer planteamiento, como escenario habitual de trabajo, para posteriormente definir una alternativa con la que se establecerán las correspondientes comparaciones. En concreto, mediante la herramienta de *VirtualBox* se prepara un entorno de red virtual con 2 máquinas virtuales sobre las que se realiza un proceso de *pentesting* típico. A continuación, se emula dicho laboratorio mediante su equivalente basado en contenedores de *Docker*. Ambas configuraciones son analizadas desde el punto de vista de rendimiento de las máquinas anfitrionas, para comprobar hasta qué punto es más rentable (computacionalmente hablando) utilizar *Docker* frente a las VMs, así como definir un ejemplo de aplicación en el ámbito educativo.

4.1 ESPECIFICACIONES PREVIAS

Al ser un trabajo que plantea un análisis comparativo de dos técnicas diferentes para la obtención del mismo servicio, es necesario poder montar todos los programas que requiere el trabajo, para lo cual es importante conocer con qué recursos físicos se ha planteado todo el estudio.

El trabajo ha sido realizado sobre un ordenador de sobremesa, con sistema operativo Windows 10 pro, un procesador AMD *Ryzen* 7 de 3,2 Ghz, una memoria RAM de 16 Gb, un almacenamiento de 250 GB SSD y 2 Tb HDD, así como una tarjeta gráfica *Gygabite* 1060 de 6Gb. Estas características no son estrictamente necesarias para la replicación de todos los pasos realizados durante el trabajo, por lo que equipos más económicos podrían ser totalmente válidos, ya que éste no fue un objetivo planteado de inicio.

Tal como se ha adelantado anteriormente, la idea principal del proyecto es utilizar como referencia un laboratorio de *pentesting*, para lo cual es necesario crear dos máquinas virtuales, cuyos sistemas son:

- **Kali**: La primera máquina (atacante) posee el sistema operativo Kali, el cual es una variante de *Debian*, preparada para entornos de hacking. Es una de la más versátiles y completas, además de la más utilizada, y parte de unos requerimientos básicos de 2 Gb de RAM y 77 Gb de memoria física, suficientes para ejecutar todas sus funcionalidades necesarias para completar el estudio de vulnerabilidades.

- **Mestasploitable**: La segunda máquina (víctima) posee una distribución de Linux, basada en versiones más o menos antiguas, puesto que es utilizada como

una mera herramienta cuyo único objetivo es ser muy vulnerable a cualquier ataque que le quiera realizar. También es una de las más utilizadas y parte de unos requerimientos básicos de 1 Gb de RAM y 8 Gb de memoria física.

Para la realización de la segunda parte del trabajo, en la que entra en juego *Docker*, se decidió implementar el sistema de *contenerización* sobre una máquina virtual, y no sobre el propio sistema anfitrión. Esto fue debido a que, por un lado. *Docker* sobre Windows requiere utilizar *HyperV* como sistema de virtualización, con características muy diferentes respecto de *VirtualBox*. Puesto que lo que se quiere comparar es el comportamiento de las máquinas virtuales frente a los contenedores, parece lógico tomar como referencia el mismo sistema de virtualización, reduciendo todo lo posible las diferencias de ejecución de los sistemas gestores. El sistema operativo de la máquina gestora (*Docker*) es un Ubuntu 18.04 LTS, puesto que *Docker* es una aplicación nativa de Linux, otra razón para evitar hacer uso de *HyperV*, que no es directamente compatible con Linux. A dicha máquina, para evitar introducir un cuello de botella en los recursos si se compara con el equipo anfitrión, se le ha dado un alto porcentaje de los recursos, para emularlo al comportamiento del propio anfitrión. En principio, cuenta con 8 Gb de RAM y 100 Gb de memoria física aproximadamente.

4.2 ESCENARIO DE APLICACIÓN

En este apartado se especifica el entorno más habitual para el desarrollo de una auditoría de seguridad, en el cual un ordenador externo, el del atacante, accede a la red de la víctima y le permite examinar las potenciales vulnerabilidades de las que se podría aprovechar para la realización de un posible ataque, como refleja la figura 6.

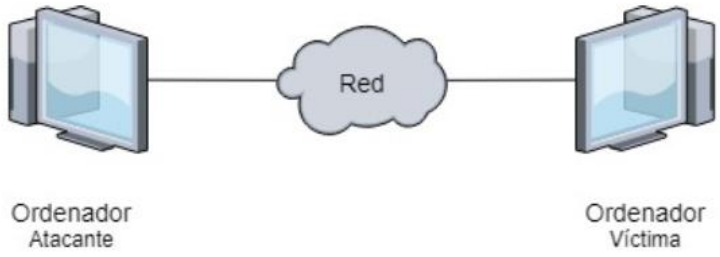

Figura 6: Esquema habitual de Pentesting

Si bien la configuración mostrada se desarrolla sobre elementos físicos, en este trabajo se parte de la necesidad de realizar una emulación de dicho escenario, mediante entornos de virtualización, bien mediante máquinas virtuales, o bien mediante contenedores, tal como se expone en los siguientes apartados.

Para llevar a cabo un análisis de Pentesting mediante este método, es necesario realizar un análisis de los equipos disponibles en la infraestructura del cliente, principalmente ordenadores, servidores y la red privada en sí. Para este proceso se recurre a las mismas fases previamente mencionadas, una fase de análisis previo, el

análisis de las vulnerabilidades, la explotación de dichas vulnerabilidades y la post-explotación. Todas estas fases son estándar en cualquier caso de Pentesting.

Si bien hasta el momento este método ha sido el más utilizado por las empresas, las auditorías hoy en día abarcan mucho más que el escenario físico, puesto que el entorno cloud está más en auge que nunca, y la necesidad de preservar la seguridad de los servicios virtualizados es fundamental, por ello la parte más relevante del proyecto se basa en los test de penetración en este ambiente.

4.3 PENTESTING MEDIANTE MÁQUINAS VIRTUALES

Sobre el primer escenario desarrollado en esta parte del proyecto se realiza un análisis de vulnerabilidades básico mediante dos máquinas virtuales, con conexión entre ellas. Es una simulación fiel del escenario real planteado en la figura 6, salvo que ahora el ordenador atacante, el ordenador víctima y la red de interconexión, como si de un ataque real se tratara. Estas 2 máquinas están alojadas en el mismo ordenador anfitrión como se refleja el esquema de red de la figura 7. A continuación se hace un resumen de las actividades que se llevan a cabo en cada fase, sin entrar en detalle en cada paso y comando, que pueden consultarse en el Anexo A. Esta descripción es usada como guía orientativa en las prácticas académicas que se realizan actualmente durante el curso [49].

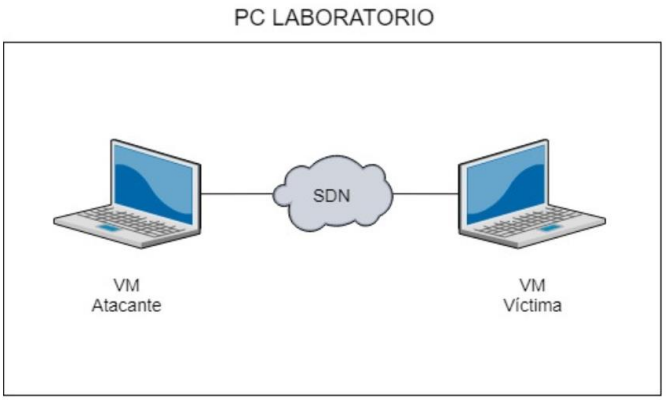

Figura 7: Esquema de Pentesting con VMs

Para esta primera versión de la auditoría de seguridad se utilizan las dos máquinas virtuales ya comentadas, en primer lugar, el atacante (*Kali*) y en segundo lugar la víctima (*Metasploitable*). Ambas están montadas sobre la plataforma de *VirtualBox*, y. ya una vez estén instaladas y configuradas, como se muestra en la figura 8, se da comienzo al test como tal.

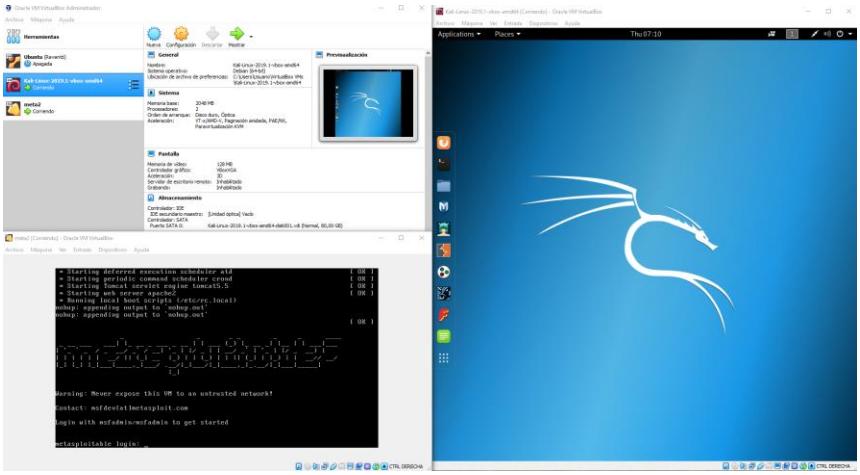

Figura 8: Máquinas virtuales instaladas y operativas

Un primer paso suele ser el hacer un escaneo de puertos con una herramienta muy utilizada para ello, **NMAP**. NMAP es una aplicación de código abierto, para el escaneo de redes. Esto permite identificar qué servicios se están ejecutando en un dispositivo remoto, así como la identificación de equipos activos, sistemas operativos, existencia de filtros o firewalls, entre otros [50]. Con esta herramienta se obtiene un primer análisis que permite hacerse una idea de cómo es la víctima, puertos abiertos y algunos detalles más, como muestra la figura 9

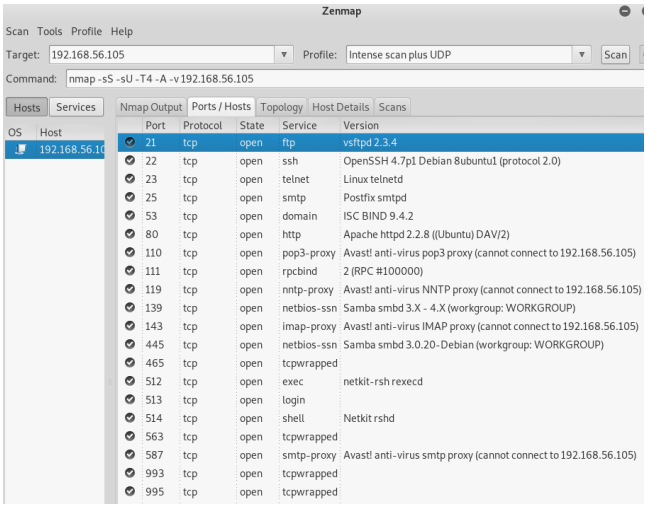

Figura 9 : Interfaz de NMAP con algunos de los puertos descubiertos

Una vez hecha esta prueba de contacto, es buen momento para iniciar el análisis de vulnerabilidades que genera un informe con todos los tipos de ataque que se le pueden realizar a la víctima, en función de la buena seguridad que tenga dicha máquina. Este análisis se hace con la herramienta **Openvas**, un escáner de vulnerabilidades, quizá el más completo. El escáner se compone de un conjunto de más de 50.000 pruebas de vulnerabilidad, con una larga historia y sus correspondientes actualizaciones. Entre sus capacidades se incluyen pruebas no autenticadas, pruebas autenticadas, protocolos industriales, protocolos de Internet de alto y bajo nivel, ajuste de rendimiento para exploraciones a gran escala y un poderoso

lenguaje de programación interno, para implementar cualquier tipo de prueba de vulnerabilidad adicional [51]. Con esta herramienta se incluye una interfaz de usuario de *Greenbone*, los desarrolladores de *Openvas*, directamente sobre el explorador que se use. Así, de forma muy intuitiva permite hacer un análisis completo, con su correspondiente informe de resultados, como muestra la figura 10. Una vez obtenido este, ya es posible saber a qué ataques es vulnerable la máquina, y además, puede ser facilitado a otra herramienta que sea capaz de explotar alguna de las vulnerabilidades detectadas.

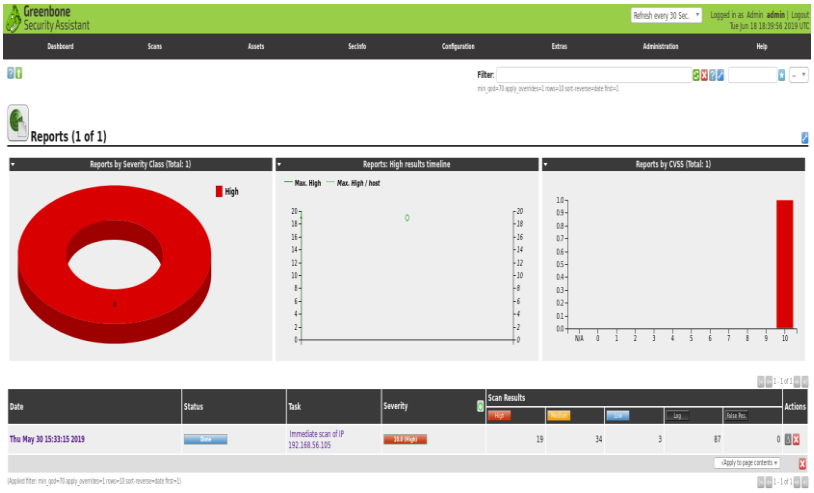

Figura 10: Página de los reportes de vulnerabilidades

La herramienta encargada de realizar los ataques es **Metasploit**, un proyecto de seguridad informática que proporciona información sobre las vulnerabilidades, ayuda en las pruebas de penetración y en la ejecución de la explotación de vulnerabilidades de seguridad. *Metasploit* representa un conjunto de herramientas que ayuda a los profesionales de seguridad y hacker a llevar a cabo ataques informáticos de manera sistematizada y automatizada [50]. Con este programa es posible importar el informe de *Openvas* y así realizar el correspondiente *exploit*, Sin embargo, presenta una interfaz de comandos, más o menos compleja, por lo que es posible apoyarse en una interfaz de usuario mucho más visual, denominada **Armitage**, una herramienta gráfica del *framework Metasploit* la cual nos permite explotar vulnerabilidades sobre cualquier equipo que esté en una red a la que se tenga acceso. Esta herramienta se puede encontrar en distribuciones de *Pentesting* como *Kali* Linux [52]. Con la interfaz gráfica de *Armitage*, se facilita el acceso a la gestión de los *exploits*, así como a las posibles ventanas de terminal de la víctima, a través de la cual revelar toda su información.

Una vez hecho un chequeo de todos los ataques posibles a la máquina, gracias al informe que se importó de *Openvas* a la base de datos de *Metasploit*, la víctima puede acabar siendo completamente vulnerable a multitud de ataques, dando lugar a una interfaz como la de la figura 11.

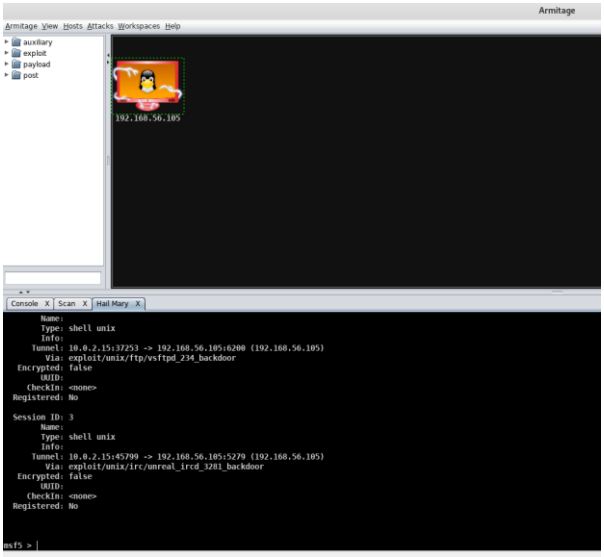

Figura 11: Interfaz de Armitage con VM vulnerable

En una auditoría de seguridad informática, tras enumerarse las vulnerabilidades, así como sus posibles soluciones, se procedería a finalizar el proceso de *pentesting*.

Hasta aquí, este sería el proceso seguido en cualquiera de las prácticas académicas realizadas, bien en máquinas físicas, o bien sobre máquinas virtuales. Para la implementación de un laboratorio completo, con múltiples puestos, el siguiente paso sería montar esta configuración dentro de un entorno Cloud, sin necesidad siquiera de que el ordenador utilizado por el alumno sea el anfitrión, ya que todas las máquinas y redes virtuales se ejecutan en la nube, tal como se describe en [49]. El esquema resultante, de forma orientativa, podría ser como el que se muestra en la figura 12.

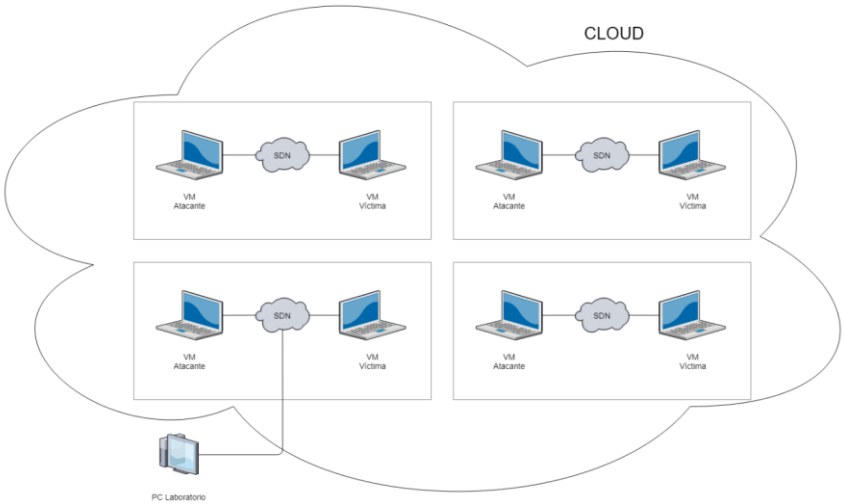

Figura 12: Esquema de Pentesting con VMs en un entorno Cloud

Según se muestra en la figura anterior, cada Laboratorio se compone de una red virtual sobre la que se despliegan la máquina atacante y víctima. Así, cada alumno accede a los terminales de cada una de las máquinas virtuales asignadas a su Laboratorio, de

forma totalmente independiente a los del resto de alumnos. Todos los componentes son ofrecidos como servicios independientes mediante un servidor de nube que es capaz de alojar todas las máquinas necesarias.

4.4 PENTESTING MEDIANTE CONTENEDORES DOCKER

De la misma forma que es posible pasar de un laboratorio con máquinas reales a otro con máquinas virtuales, también es posible desplegar los mismos componentes mediante contenedores, los cuales deberán contener los servicios necesarios para completar cada una de las fases del proceso de *pentesting*. Para ello se parte de un esquema muy similar al anterior, esta vez con dos contenedores, el atacante que replica el sistema *Kali* y la víctima, que replica al sistema *Metasploitable*. Ambos contenedores son gestionados por el *Docker* de la máquina virtual Linux, que actúa como anfitrión, tal y como se muestra en el esquema de red de la figura 13. Al igual que en el caso anterior, a continuación, se describen las fases realizadas, sin entrar en detalle en cada paso y comando, los cuales pueden encontrarse en el Anexo B, al final de este documento.

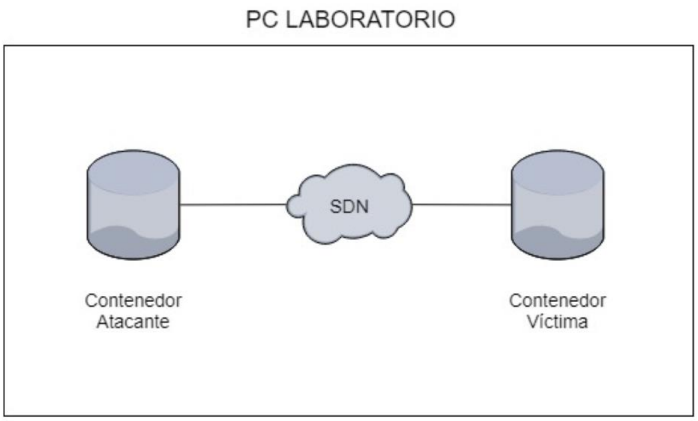

Figura 13: Esquema de Pentesting mediante contenedores

Tras preparar la máquina vitual, a modo de ordenador anfitrión, con un Linux Ubuntu, se debe instalar el programa *Docker* para poder utilizar todos los comandos necesarios.

Una vez con el entorno preparado, hay que saber elegir la imagen en la que basar el desarrollo para poder crear el contenedor más acorde a las necesidades. Al ser una tecnología tan novedosa, no hay imágenes oficiales de todos los programas que se necesitan, e incluso las imágenes oficiales existentes pueden estar muy limitadas con respecto a todos los servicios que se requieren. Esto hace que esta parte del desarrollo sea una de las menos gratificantes, ya que no solo hay que probar lo que han hecho otros usuarios de la comunidad, sino que toca experimentar, mediante ensallo y error hasta encontrar aquel contenedor que más se asemeje a lo que se busca, y añadir todas las modificaciones necesarias hasta obtener el servicio buscado.

En el caso del contendor Metasploitable, hubo numerosos problemas con la imagen elegida, debido a que durante la realización del análisis de vulnerabilidades no era posible encontrar ninguna forma de atacar al contenedor, por lo que lo hacía inservible. Tras una larga investigación y numerosas pruebas, se descubrió que el problema era que dicha imagen tenía los servicios desactivados, por lo que no se podía utilizar con el propósito deseado a menos que se activasen manualmente uno a uno. Finalmente, con fortuna se pudo encontrar una imagen alternativa realizada por un usuario que sufrió el mismo problema, esta persona conseguía activar la mayoría de los servicios del contenedor durante el arranque del mismo, como muestra la figura 14. De esta manera se consiguió un contenedor adecuado para la realización del proyecto.

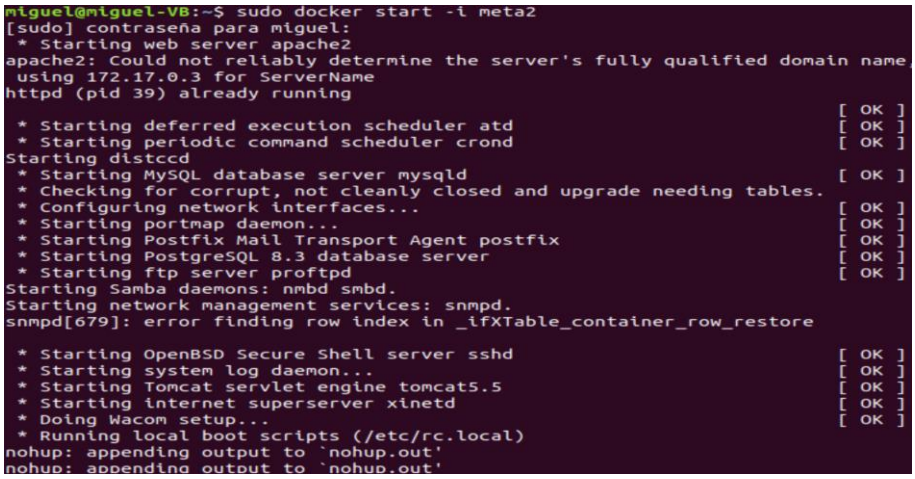

Figura 14: Ejemplo de arranque del contenedor Metasploitable

En el caso del Kali, dada la complejidad de este sistema, también surgieron muchos problemas. La imagen oficial está sumamente limitada, ya que no incluye másque algunas de las herramientas básicas y, además, genera muchos errores puesto que no está del todo optimizada para el tipo de herramientas utilizadas en el *Pentesting* elegido. Sin embargo, es muy fácil encontrar contenedores a medida, con cada uno de los programas que se necesitan, y que funcionan realmente bien. Son una solución si no se necesita tener obligatoriamente todo aunado en un solo contenedor. Sin embargo, se ha deseado conseguir el resultado más óptimo posible, así que se eligió una imagen con el programa más difícil de conseguir, *Armitage*, y que estuviese basado en *Kali*. A partir de esta base, se ha obtenido un nuevo contenedor más óptimo, aunque no sin encontrar numerosos problemas por el camino, todos ellos relacionados con *Openvas*.

Durante la instalación de Openvas, se observa cómo ésta presenta varias dependencias con otros programas, como es el caso de *Redis*, un motor de base de datos en memoria, basado en el almacenamiento en tablas de hashes, pero que opcionalmente puede ser usada como una base de datos durable o persistente [53]. Tras conseguir solucionar dichas dependencias, el contendor permite el acceso al interfaz de usuario de *Openvas*, como refleja la figura 15.

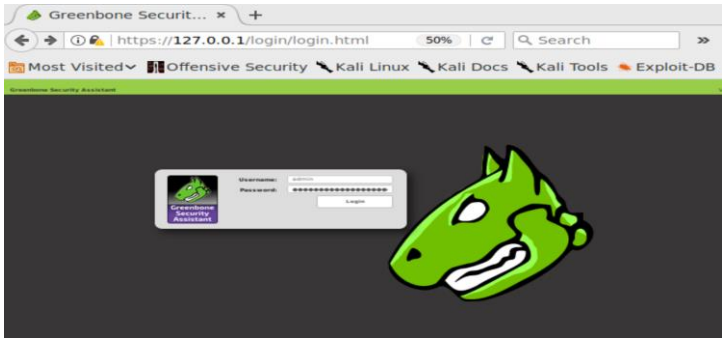

Figura 15: Interfaz de acceso de Openvas

Al igual que en el apartado anterior, se hizo el análisis de vulnerabilidades propio de *Openvas*, dando lugar a un reporte satisfactorio como muestra la figura 16. Este análisis de vulnerabilidades es un proceso que conlleva nastante tiempo, en torno a una hora en el mejor de los casos, así como un gran consumo de recursos.

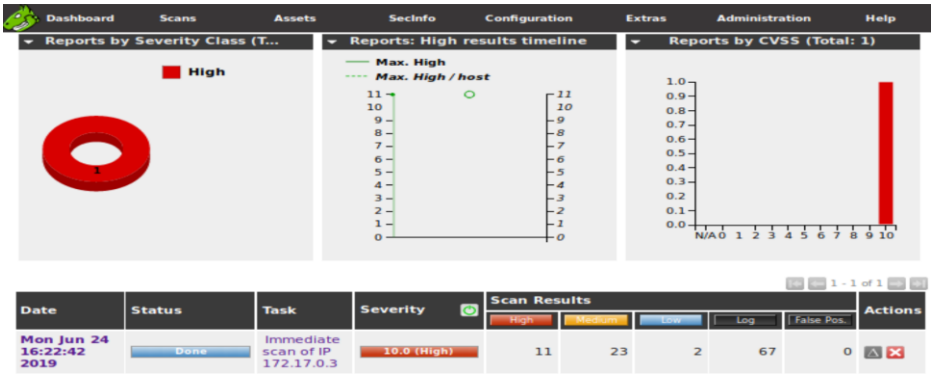

Figura 16: Página de análisis de vulnerabilidades

Este reporte mostró numerosas vulnerabilidades, de diversa gravedad, en el contenedor de la víctima, lo que significa que efectivamente la víctima era muy vulnerable a numerosos tipos de ataques. La figura 17 refleja algunas de ellas.

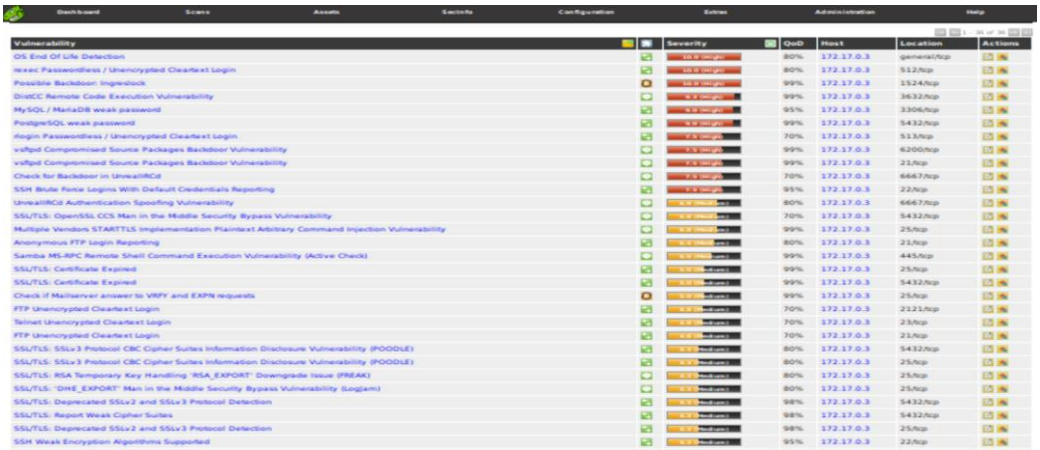

Figura 17: Listado de algunas vulm¡nerabilidades importantes

Este informe aporta un mejor análisis para la posterior explotación de las vulnerabilidades que aparecen.

Ya con el reporte obtenido, el siguiente paso es importarlo a la base de datos de *Metasploit*, lo que permite aprovecharse de las vulnerabilidades encontradas para la ejecución de los *exploits* oportunos. El acceso a *Metasploit* genera una interfaz como la de la figura 18 o similar, dispone de muchos diseños aleatorios.

Figura 18: Interfaz de inicio de Metasploit

Una vez el informe esté en la base de datos, se utilizará la aplicación *Armitage*, como interfaz gráfica de *Metasploit*, por ser una forma más cómoda y visual de poder ejecutarlos. Tras su arranque y búsqueda de la víctima, la interfaz generada debería ser como la de la figura 19.

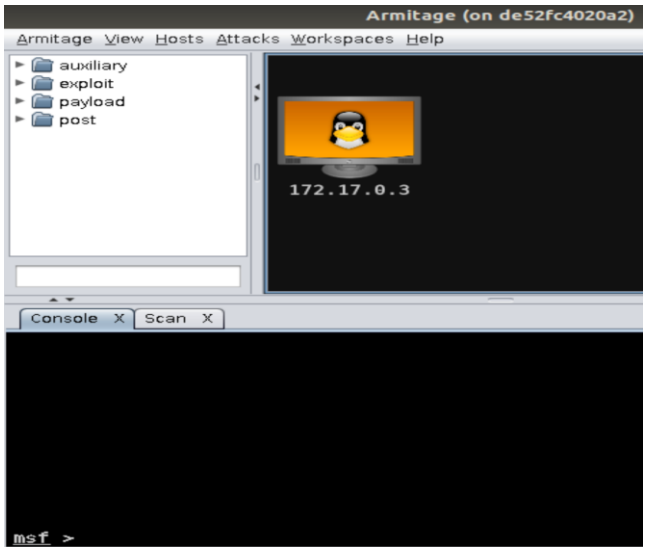

Figura 19: Interfaz de Armitage con la víctima detectada

Ya en este punto únicamente queda la explotación de las vulnerabilidades, para ello, mediante la utilización de "*Hail Mary*", un comando que explota todas las vulnerabilidades sucesivamente, la máquina se posiciona en una situación de total

vulnerabilidad, como se ve en la figura 20. En este estado es posible tomar el control total de la máquina y poder acceder a elementos tan esenciales como la terminal del equipo.

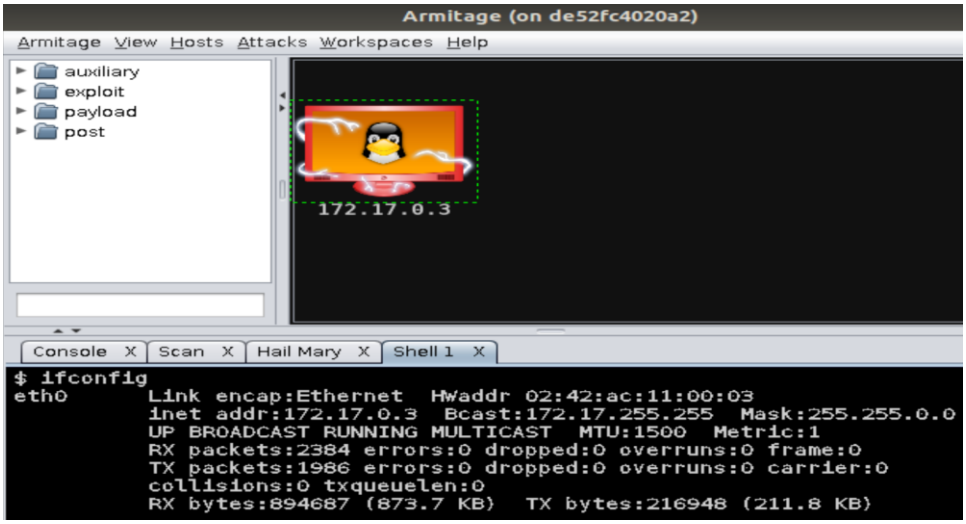

Figura 20: Situación de vulnerabilidad total en la víctima

Una vez conseguido el ataque al contenedor *Metasploitable* se habrá conseguido el objetivo que se quería, emular de la manera más exacta posible las posibilidades de *pentesting*, idénticas a las que se tenían en las máquinas virtuales.

Hay que tener en cuenta que estas herramientas están aún en desarrollo, por lo que su instalación y configuración son muy laboriosas y llenas de dificultades, pero una vez que se ha hecho un buen trabajo en este aspecto, el rendimiento es exactamente el mismo que el de una máquina virtual, e incluso física. Ahora solo queda saber si es cierto que consumen menos recursos que estas y por lo tanto son una alternativa viable en ciertas situaciones.

4.5 ANÁLISIS DE RECURSOS

En este apartado se aborda el análisis de los recursos computacionales requeridos por los dos casos presentados anteriormente. El principal objetivo es probar si son ciertas las numerosas ventajas de rendimiento que presenta la tecnología de contenedores frente a la de virtualización. Para ello, el análisis se centrará en la máquina virtual y su correspondiente contenedor *Kali Linux*, ya que éste realiza las operaciones más exigentes, computacionalente hablando, mientras que *Metasploitable* utiliza siempre las funciones mínimas en ambos casos y la diferencia es muy poco apreciable.

Este análisis tiene en cuenta los tres recursos más relevantes para el análisis de cualquier software, la memoria RAM, el uso de la CPU, y el tamaño de memoria física ocupado. Para ello se siguen dos métodos similares que permiten medir ambos casos de la manera más exacta posible. En el caso de la máquina virtual, se hace uso del monitor de recursos propio de *Debian*, lo que permite monitorizar el porcentaje de

CPU que utiliza y el consumo de memoria RAM. Así, por ejemplo, la figura 21 muestra una captura en un momento de reposo de la máquina, en la que se puede apreciar que, el mero funcionamiento del sistema operativo, requiere cierto porcentaje de CPU y un consumo mínimo de 1.1 GB de RAM de los 2GB destinados a la máquina.

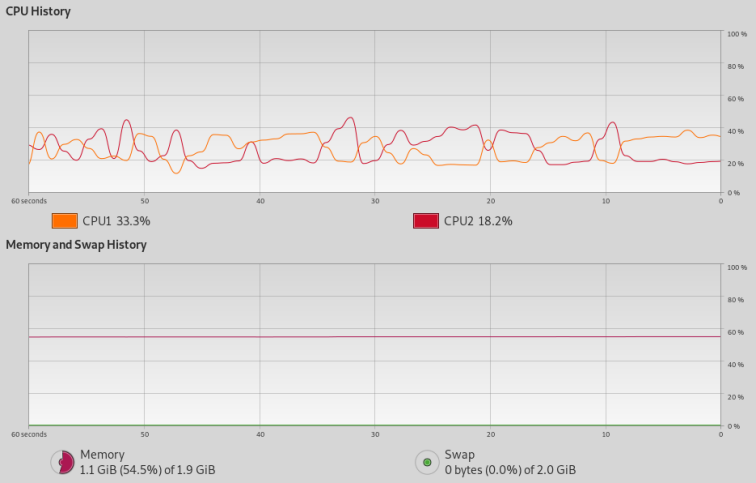

Figura 21: Consumo de CPU y RAM de VM Kali

Para obtener cuánta memoria física está ocupada, es decir, disco,existe un comando "*df –h*", que permite saber la cantidad de memoria total destinada a la máquina y cuánta se está utilizando en ese momento. En este caso en particular, que refleja la figura 22, la opción *"/dev/sda1"* muestra que la máquina dispone de 77 GB y utiliza 15 GB, es decir, un 20% de la memoria.

```
root@kali:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            977M     0  977M   0% /dev
tmpfs           200M  6.3M  194M   4% /run
/dev/sda1        77G   15G   59G  20% /
tmpfs           998M     0  998M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           998M     0  998M   0% /sys/fs/cgroup
tmpfs           200M   16K  200M   1% /run/user/130
tmpfs           200M   28K  200M   1% /run/user/0
```

Figura 22: Memoria total de la máquina (77Gb) y memoria usada (15Gb)

En el caso de los contenedores, la situación es un poco diferente debido a que el monitor de recursos del sistema operativo que los contiene no los considera procesos en sí, por lo que no es capaz de registrarlos. En su defecto, el programa *Docker* cuenta con un comando, "*docker stats*", capaz de monitorizar los consumos previamente mencionados en tiempo real. Al igual que se hizo en el caso anterior, se hizo captura de un momento en el que el contenedor se encontraba en reposo, dando lugar a los resultados de la figura 23. En ella se ve como los consumos de CPU (4,47%) son inmensamente inferiores a los de la máquina virtual, debido a que no necesita albergar un sistema operativo completo, sino que en su lugar solo cuenta con las funciones necesarias. Por su psrte, en el caso de la memoria RAM, la situación es similar, el contenedor en reposo consume el 8% de la memoria RAM de la máquina en la que se aloja, que es de 8 GB, por lo que se encuentra consumiendo 0,64 GB, casi la mitad de

lo que requiere la máquina virtual. La principal ventaja estriba en que, el contenedor reserva memoria de forma dinámica, es decir, toma lo que necesita sin necesidad de tener reservada una cantidad de memoria fija como en el caso de la máquina virtual. Para terminar, en el caso de la memoria física, los contenedores por defecto reservan 7,79 GB de memoria, pero estos se pueden reducir o aumentar sin problemas. Aún así, en este caso, de todo lo que dispone únicamente está utilizando 638,5 MB, muy lejos de los 15 GB que requería el sistema operativo de la máquina.

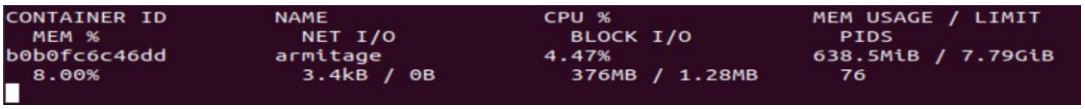

Figura 23: Consumos de RAM, CPU y memoria física del contenedor Kali

La gráfica de la figura 24 muestra, a modo de resumen, las sustanciales diferencias que se han mencionado previamente.

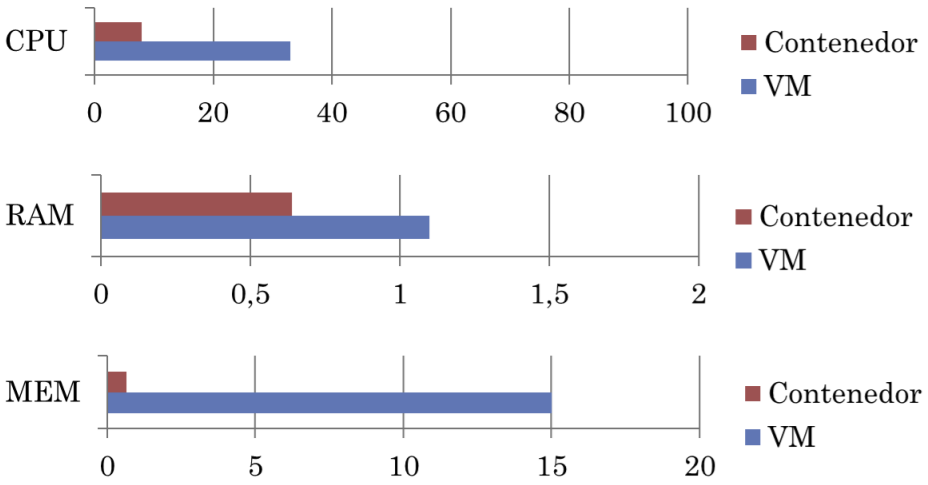

Figura 24: Gráfica de consumo de recursos en reposo

Sin embargo, los resultados anteriores se corresponden con períodos de reposo, por lo que también se han analizadolos consumos de CPU y memoria RAM durante la realización de un proceso exigente, como es el caso del análisis de vulnerabilidades de *Openvas*. Este es uno de los procesos que más recursos requiere en un primer momento. Además, no se ha tenido en cuenta el valor de la memoria física puesto que la variación sigue siendo muy leve.

En primer lugar, al igual que en el estudio anterior, se analiza la máquina virtual, dando lugar a unos incrementos en los recursos bastante significativos, como muestra la figura 25. En ella podemos ver como los consumos de CPU crecen considerablemente y se mantienen bastante altos. Por otro lado, la memoria RAM llega casi al máximo de lo que tiene reservado.

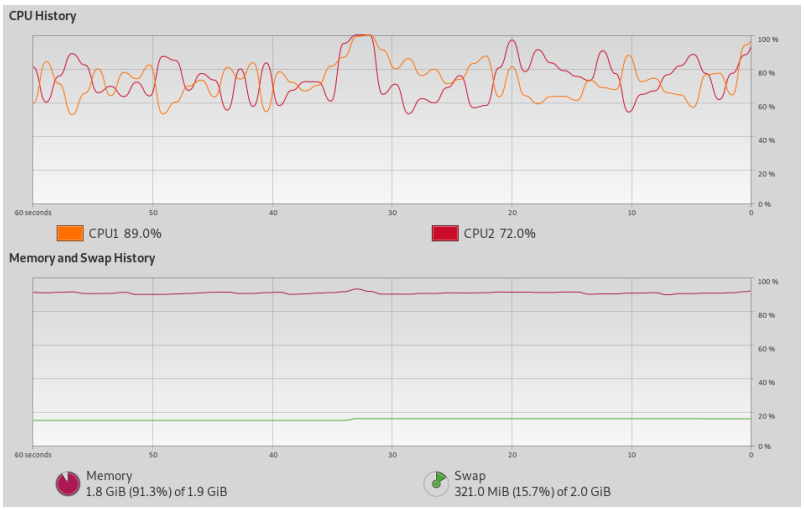

Figura 25: Consumo de CPU y RAM en VM Kali durante el análisis

En el caso de los contenedores también aumenta su consumo como sucede con las máquinas, aunque en menor medida que estas. En la figura 26 se puede ver los resultados extraidos, según los cuales, la memoria RAM llega a requeris hasta un 20%, es decir, 1,6 Gb de la memoria del anfitrión. Sin embargo, este aumento tan notable es debido al resto de herramientas includias además del Openvas, que durante su instalación realizan su propia gestión de memoria para poder funcionar dentro del contenedor. Probablemente, con un diseño más eficiente y pulido, se podría haber conseguido un contenedor más eficiente, sin incluir dichas herramientas, pero aun así, es mucho los resultados de la máquina virtual. Por su parte, al centrar la atención en el consumo de CPU, se observa todavía mejor la diferencia, ya que, en todo momento, e contenedor e mantiene con un consumo muy bajo, comparado con el de la máquina virtual, aunque ligeramente superior que en reposo. Además, en el caso de que puntualmente se necesitase un mayor uso de memoria ante un proceso de altos requerimientos, habría que sumar el hecho de que a la máquina virtual se le asigna una cantidad de memoria RAM determinada durante su creación, por lo que está limitada y, aunque lo necesitase, no puede utilizar más RAM por su cuenta.

```
CONTAINER ID        NAME                CPU %
  MEM %                 NET I/O             BLOCK I/O
de52fc4020a2        armitage2           13.70%
  20.12%                327MB / 3.63MB      6.16GB / 6.63GB
```

Figura 26: Consumo de RAM y CPU del contenedor Kali durante el análisis

En la gráfica de la figura 27 se muestra a modo de resumen la diferencia de consumos durante la ejecución.

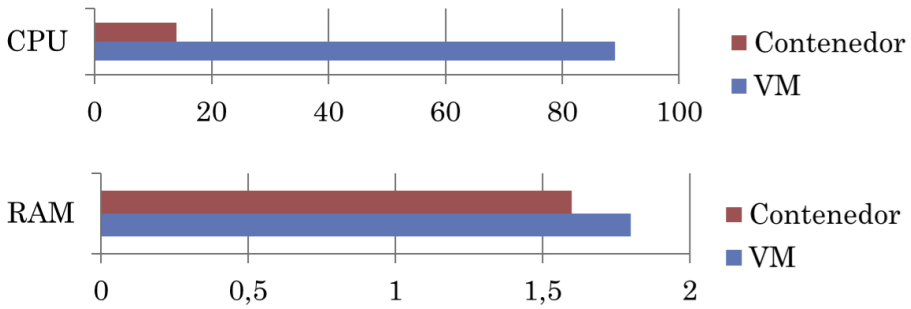

Figura 27: Gráfica de consumo de recursos en ejecución

Como conclusión a este análisis de recursos, se puede extraer que efectivamente las ventajas que se atribuyen a los contenedores con respecto a la máquinas virtuales son ciertas. Êstas herramientas no necesitan de un sistema operativo propio que los contenga, sino que aprovechan el del Host, y tienen la capacidad de utilizar los recursos justos que necesitan, asignándolos de forma dinámica. Estohace que sean mucho más eficientes en cuanto a consumos, por lo que en numerosas aplicaciones son una alternativa perfectamente viable.

4.6 PRUEBA DE CONCEPTO DE PENTESTING PARA SU APLICACIÓN EN LOS LABORATORIOS DOCENTES

En este último apartado, se presenta una propuesta de aplicación real en el entorno académico a esta tecnología. Por ello se plantea cómo podría llevarse a cabo una práctica docente en asignaturas de seguridad o que traten temas de virtualización. Todos los detalles sobre los comandos necesarios para las conexiones necesarias se especifican en el Anexo C. En concreto, se plantea la realización de dos tipos de prácticas, la primera más enfocada al aprendizaje sobre el *Pentesting* y la segunda dirigida hacia el aprendizaje de los contenedores. Sea cual sea la opción elegida el esquema a seguir sería similar al de la figura 28.

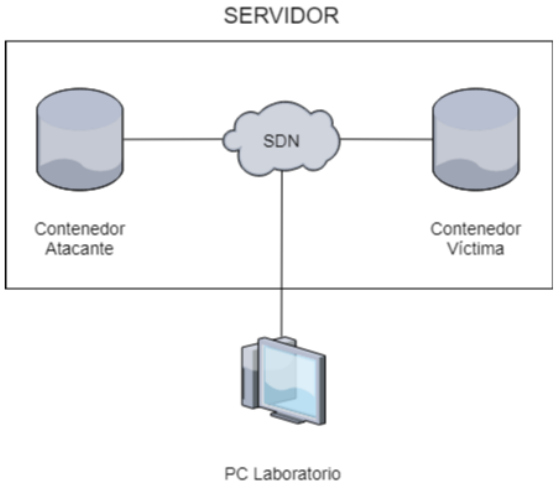

Figura 28: Esquema de Pentesting en laboratorio docente

Para la realización de la primera práctica (*Pentesting*), se parte de los contenedores ya creados y alojados, por el profesor responsable, en el servidor de virtualización del Laboratorio Docente. De esta manera, los alumnos, conociendo únicamente la dirección IP del contenedor atacante que desean controlar, podrían establecer una conexión **SSH** que les permita tomar el control de dicha máquina. SSH es el nombre de un protocolo y del programa que lo implementa cuya principal función es el acceso remoto a un servidor por medio de un canal seguro en el que toda la información está cifrada [54]. Para el establecimiento de dicha conexión se sugiere utilizar una herramienta llamada **Putty**, una implementación gratuita que permite realizar esta y otros muchos tipos de conexiones. La interfaz de esta herramienta se muestra en la figura 29.

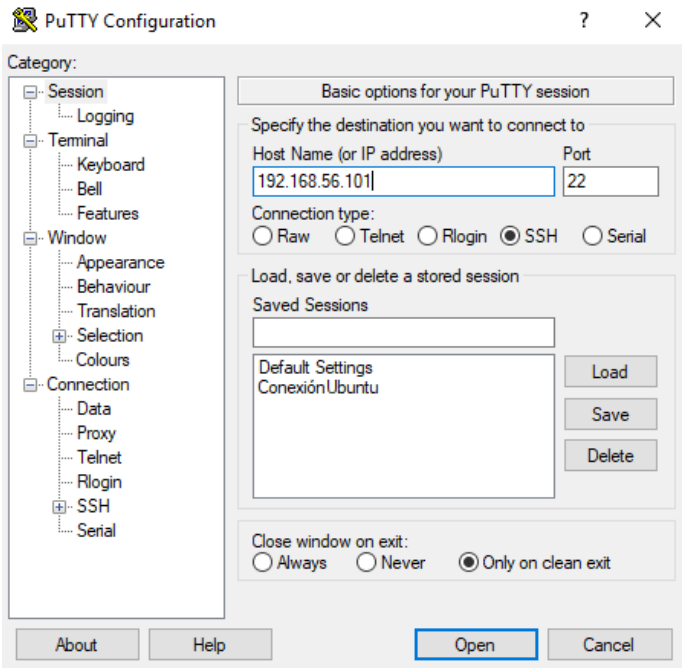

Figura 29: Interfaz de Putty

En esta herramienta únicamente con facilitar la IP del equipo al que se quiere controlar, el usuario y contraseña de dicho equipo, es suficiente para tener el control de un terminal de dicho equipo, como se observa en la figura 30.

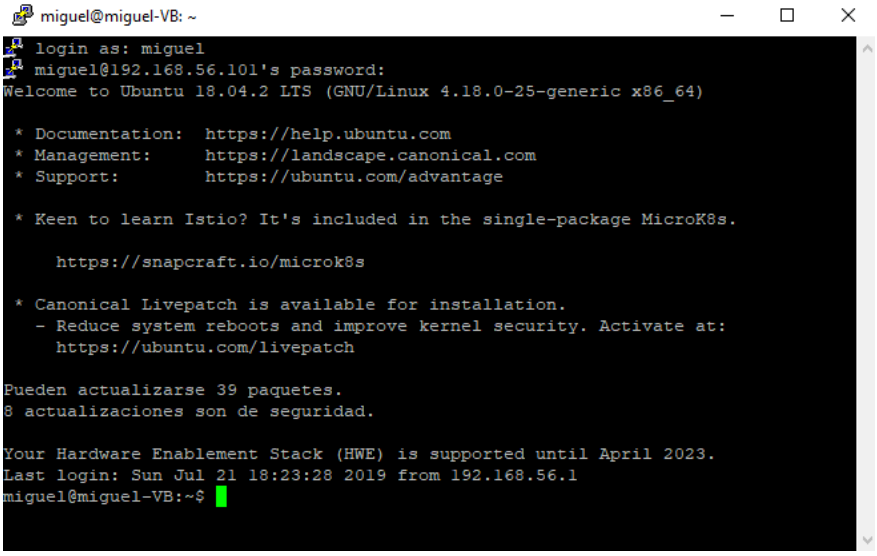

Figura 30: Conexión SSH establecida con éxito

Ya una vez tomado el control del atacante, solo es necesario conocer la dirección IP de la víctima a la que atacar. El profesor puede facilitar dicha IP, o bien el alumno debe utilizar alguna de las herramientas incluidas en su máquina atacante, como por ejemplo el escaneo de puertos mediante **NMAP**. A continuación, la práctica puede consistir en la realización de un *pentesting* completo, teniendo en cuenta algunos pasos especificados en el Anexo C, especialmente todos aquellos relacionados con el acceso a algunas funciones con interfaz gráfica. Teóricamente, funcionando todo de manera correcta, los alumnos son capaces de realizar todos los pasos del *pentesting*, esta vez sobre contenedores. Esto incluye desde el análisis de vulnerabilidades con *Openvas*, hasta la explotación con *Metasploit*. Esta práctica puede generar múltiples variante, como por ejemplo, que todos los puestos atacasen a una misma víctima para así obtener los mismos resultados y poder analizarlos de una manera más sencilla en clase.

Para la realización de la segunda posible práctica, la idea es que los alumnos sean, los encargados de hacer todo el proceso de instalación de los contenedores, incluso en algún paso durante su creación. Para ello, a partir de las imágenes que pueda facilitarles el profesor, los alumnos tienen que ser capaces de crear los contenedores y alojarlos en la red que deseen, sin perder el acceso a ella. De esta forma es posible hacer todos los pasos previos de la auditoría de seguridad y ser conscientes de las dificultades que implica una instalación completa de *pentesting* sobre contenedores hoy en día, puesto que es una tecnología que no está depurada del todo.

5 CONCLUSIONES Y LÍNEAS FUTURAS

Las TIC hoy en día son imprescindibles tanto para las empresas como para cualquier usuario de a pie, esta tecnología ha ofrecido muchas ventajas en numerosos ámbitos, a nivel informativo, económico, comunicativo, entre otro muchos.

No obstante, no todo son ventajas, como en todo, donde hay algo rentable está también algo ilegal de lo que sacar provecho, por eso los profesionales informáticos recurren a las auditorías de seguridad informática para realizar análisis que aseguren la fiabilidad de la infraestructura de red de las empresas.

Así mismo, en el ámbito de la virtualización lo común es utilizar máquinas virtuales que permitan tener uno o varios equipos virtuales en un mismo equipo anfitrión, pero pese a ello, están surgiendo alternativas y mejoras a esta tecnología. El uso de contenedores *Docker* es una de ellas y debe ser tenida muy en cuenta debido a que, en un futuro, es muy probable que sean mucho más comunes y utilizados de lo que se piensa, ya que ofrecen prácticamente las mismas ventajas que las máquinas virtuales, pero de una forma mucho más eficiente.

En este trabajo, se ha logrado crear un laboratorio de *pentesting* mediante la tecnología de contenedores, totalmente operativo, y que puede ser utilizado en entornos académicos, para la implementación de prácticas docentes en los laboratorios de seguridad y/o virtualización.

La implementación de un laboratorio idéntico, pero esta vez basado en máquinas virtuales, ha permitido realizar una comparativa entre ambas soluciones. Los primeros resultados muestran que la simulación mediante contenedores supone un gran ahorro de recursos en el equipo anfitrión, por lo que el número de aplicaciones y servicios soportados por el mismo se ve automáticamente incrementado. Ha esto hay que añadir las grandes posibilidades de portabilidad e implantación en casi cualquier equipo de un contenedor. Si a esto se añade la posibilidad de aplicar técnicas de orquestación sobre dichos contenedores, el despliegue de los mismos se flexibiliza hasta puntos inalcanzables por las tecnologías de máquina virtual.

Además, a modo de aplicación práctica del estudio, se ha propuesto un par de ideas para su aplicación en las prácticas académicas de Ingeniería de Telecomunicaciones de la Universidad de Cantabria. Tanto la seguridad, como la virtualización, son temas de máximo interés, no solo en la rama de telemática, y este tipo de despliegues pueden ser exportados hacia otras asignaturas del área, o incluso a otro tipo de contenidos

Desde el punto de vista de las mejoras que podrían incluirse en un futuro cercano, conseguir optimizar los contenedores desarrollados, especialmente el contenedor *Kali* es uno de los más urgentes. Es realmente interesante reducir aún más el consumo de recursos. Así mismo, es interesante poder incluir librerías más eficientes, que permitan conseguir crear los contenedores deseados de una manera más rápida. Por otro lado, resulta especialmente interesante conocer cómo se puede implementar una orquestación en los contenedores, posiblemente mediante la aplicación de

herramientas como *Kubernetes*, y que permitan una mayor gestión de todo el ecosistema así generado.

Aunque ha sido mencionado anteriormente, la migración de estos sistemas a un entorno de Cloud es realmente interesante. Así, *OpenStack*, el sistema de Cloud que se está desplegando en los Laboratorios del Grupo de Ingeniería Telemática, ofrece muchas posibilidades de cara al futuro, puesto que los recursos los albergan dos servidores independientes que ofrecen una red virtual que puede ser conectada desde cualquier equipo. Esta idea se refleja en la figura 31.

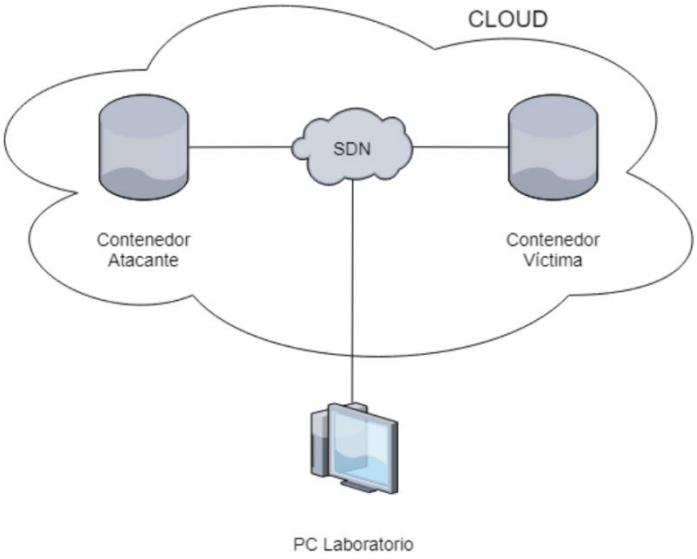

Figura 31: Contenedores alojados sobre entorno cloud

Por último, desde el punto de vista de la seguridad, estos entornos basados en contenedores facilitan la idea de probar otros tipos de auditorías de seguridad informática, con otro tipo de atacantes o víctimas, Esto permitiría ampliar conocimientos acerca de ellas, así como extraer conceptos más generales sobre cómo se llevan a cabo. Además, el hecho de mantener todos los recursos en entornos de nube, facilita el poder realizar un escenario de *pentesting* a mayor escala, donde todos los recursos se encuentren en la nube, pero que a su vez pudiesen conectarse desde diferentes redes, y combinen contenedores con máquinas virtuales, sería el escenario más desarrollado, cara a las necesidades que tuviese cada uno. El esquema que refleja la idea se muestra en la figura 32.

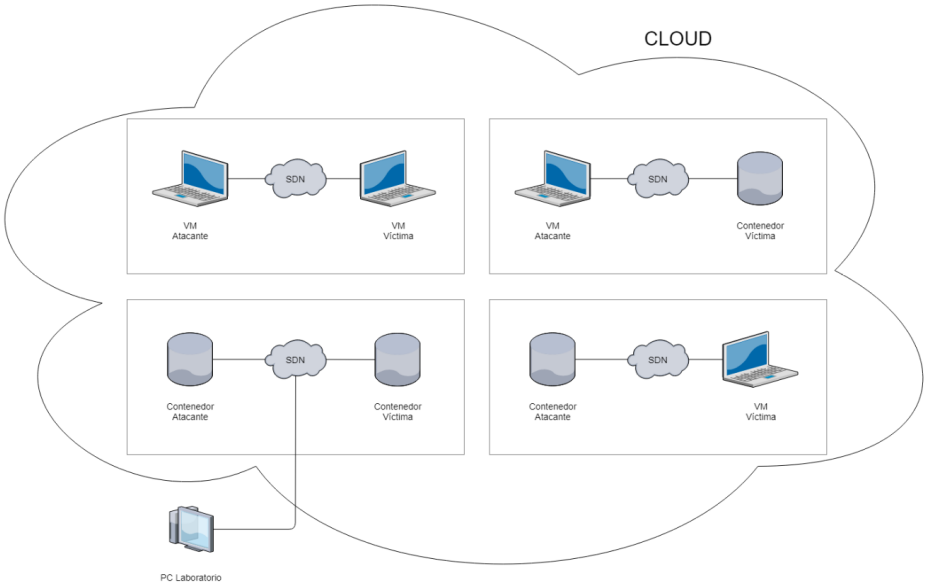

Figura 32: Escenario completo de pentesting sobre cloud

6 ANEXOS

En esta parte del proyecto se detallará cada una de las instalaciones de los programas, así como los comandos utilizados para su correcto funcionamiento.

6.1 ANEXO A

En este caso, se explicará paso a paso cada fase, sin entrar en detalle al porqué, debido a que ya se detalló previamente en el punto 4.2 PENTESTING MEDIANTE MÁQUINAS VIRTUALES.

En primer lugar habrá que descargar e instalar el Kali Linux, para ello se accederá a la página oficial de kali [55] la cual redirige a otra web [56], mostrando una página como la de la figura 33, en la cual se podrá bajar la versión de 64 bits, en formato .ova, que se necesitará para VirtualBox

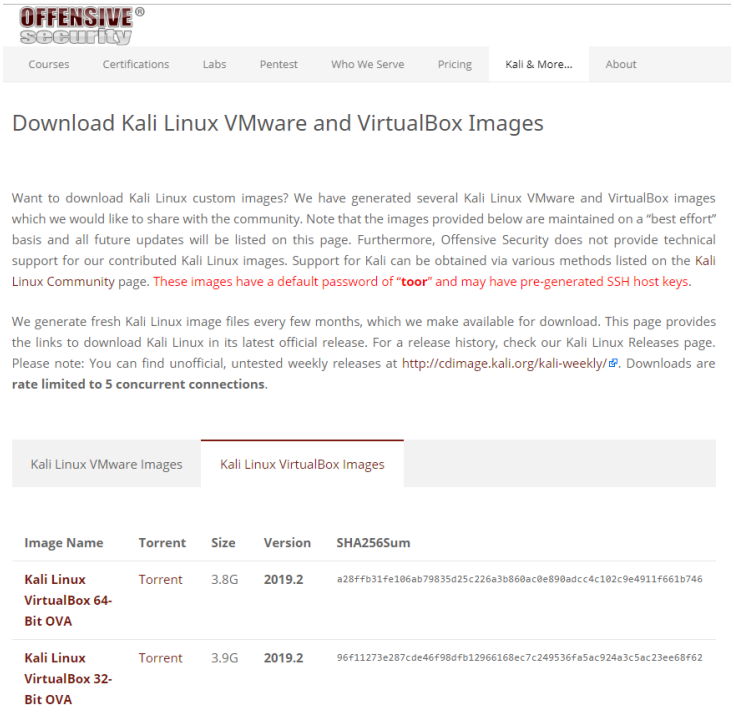

Figura 33: Página de descarga de Kali para VM

Ya teniendo descargado el archivo hay que abrir el programa VirtualBox, el cual presentará una interfaz como la de la figura 34, donde se procederá a la creación de la máquina.

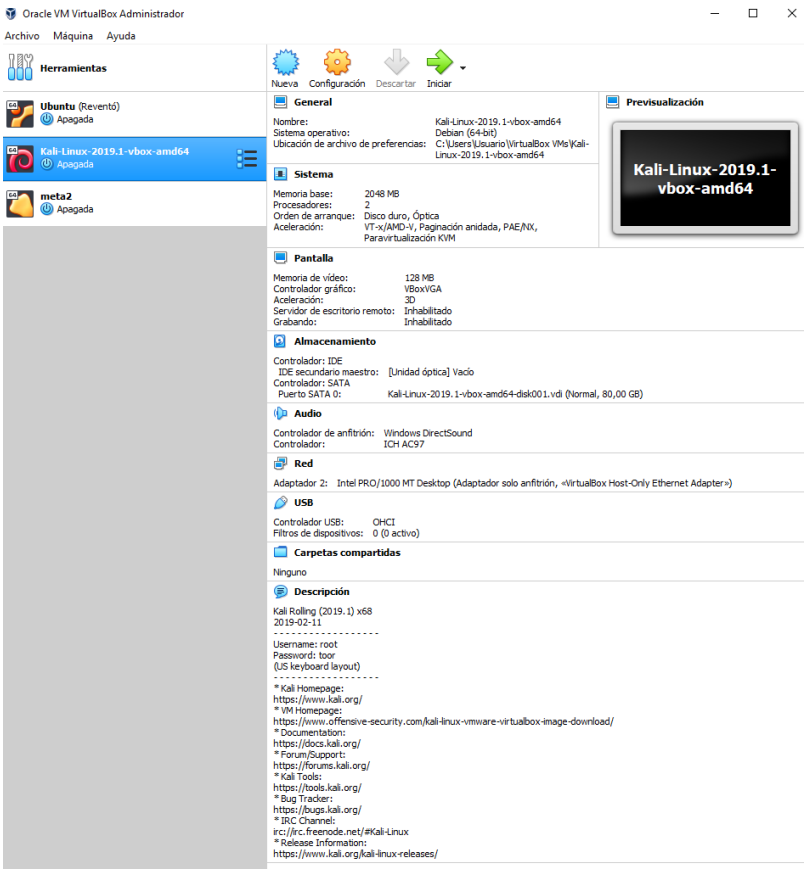

Figura 34: Interfaz de VirtualBox

Los pasos a seguir para la instalación son:
1. Seleccionar “Archivo”.
2. “Importar servicio virtualizado”.
3. Buscar la ubicación en la que se encuentra el archivo .ova descargado y pulsar “Next”.
4. Elegir la ubicación donde lo queremos guardar y pulsar “Importar”.

Estos pasos crearán una máquina funcional con los requisitos mínimos para que funcione debidamente, los cuales son un uso de la memoria RAM de 2Gb y un uso de la memoria interna del dispositivo de 77Gb.
Previamente a ser arrancada la máquina será necesario configurar el apartado de Red, creando un nuevo adaptador de red en modo “Host-Only” como muestra la de la figura 35, que permita realizar los mapeos de puertos y los ataques de forma segura, aunque para las instalaciones será necesario seguir utilizando el modo “NAT” que viene por defecto.

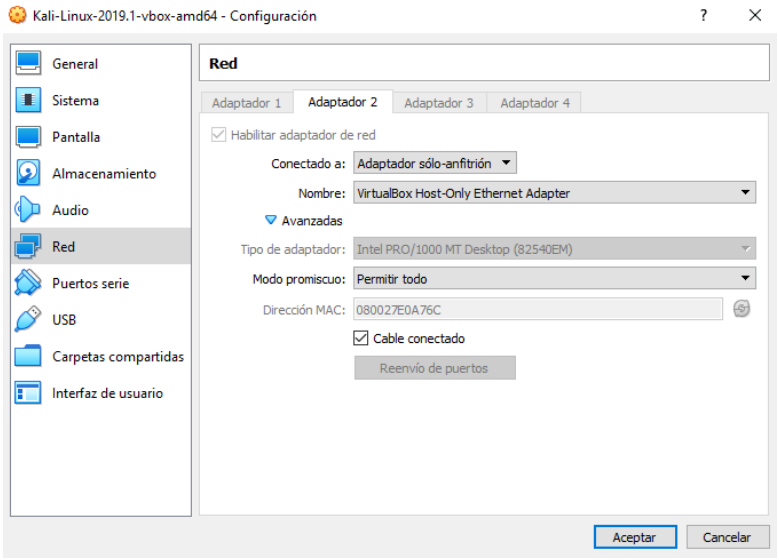

Figura 35: Configuración de la red

Así mismo será necesario modificar un defecto de la instalación en el apartado USB que impide el arranque de la máquina si permanece en la opción 2.0, para ello será aconsejable deshabilitarlo o cambiarlo a la opción 1.0.

Con todos estos pasos ya realizados, la máquina ya está preparada para su inicio, por lo que se procederá a la instalación del Metasploitable. La descarga de este archivo se puede realizar desde diferentes webs, pero se eligió [57] por ser intuitiva y clara con las explicaciones. Está página nos redirigirá a [58], donde tras un breve registro con nuestros datos permitirán descargar dicho archivo, el cual en este caso será de tipo .vmdk.

Una vez ya con el archivo se procedes a la instalación de la máquina, para ello el proceso es algo diferente al de Kali, puesto que los pasos a seguir son:
1. Seleccionar “Nuevo”.
2. Elegir nombre de la máquina (Meta2), ubicación de la máquina y sistema operativo así como la versión (Otros Linux).
3. Asignar cuánta memoria RAM queremos destinar a la máquina (1Gb).
4. Crear disco duro virtual.
5. VDI (Virtual Disk Image).
6. Reservado dinámicamente
7. Proporcionamos memoria a la máquina (8Gb)

Ya una vez con la máquina creada habrá que hacer un paso muy importante para evitar un problema de acceso que impediría atacar esta máquina, para ello hay que mover el archivo .vmdk descargado a la carpeta que se ha creado en VirtualBox al generar la máquina. Una vez colocado, se accederá a la opción de almacenamiento de la máquina y se asignará el archivo .vmdk como el predefinido al que acceder para arrancarla, como aparece en la figura 36.

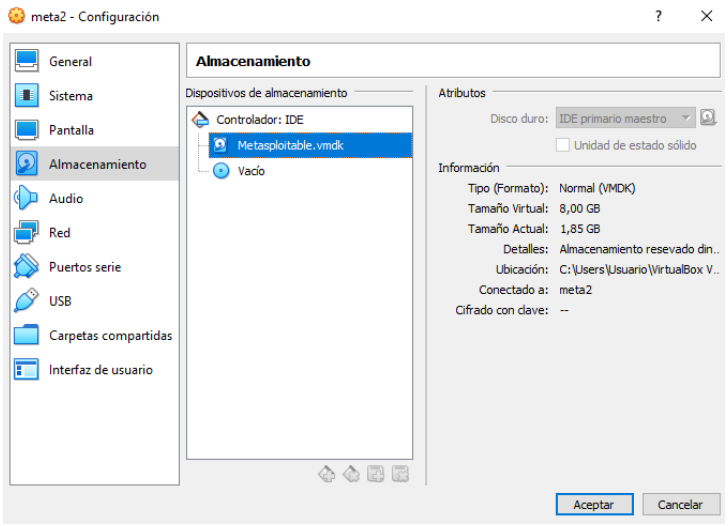

Figura 36: Interfaz donde cargar el archivo.vmdk

Finalmente al igual que se hizo en la máquina Kali será necesario crear la nueva red "Host-Only", la misma que la de Kali para que se comuniquen, así como desactivar la opción de USB. Ya en este punto ambas máquinas están configuradas y preparadas para su arranque.

En el caso del Metasploitable únicamente hay que arrancarlo y cuando pida el usuario y la contraseña, ambos son "msfadmin", hay que hacer un "ifconfig" para saber su dirección ip de la red "Host-Only" y ya estará preparada para ser atacada.

En el caso de Kali Linux, tras su arranque, las claves de acceso son "root" para el usuario y "toor" para la contraseña. Una vez dentro se podrá utilizar ZNMAP (NMAP) para efectuar el rastreo de puertos indicando la ip de la víctima (192.168.56.105) y el tipo de escaneo que se quiere realizar (Intense scan plus UDP), tal y como se ve en la figura 20. Como resultado ofrece un listado de puertos libres por los que acceder a dicha máquina así como algunos otros resultados menos relevantes para este proyecto.

Una vez ya hecha la toma de contacto, se procederá a la instalación de Openvas, para ello hay que introducir una serie de comandos en un terminal de la máquina:

$ apt-get update && apt-get dist-upgrade

Este comando es simplemente una recomendación para tener la máquina con la última actualización.

$ apt-get update && apt-get install -y openvas

Instalación y actualización de los servicios de Openvas.

$ openvas-setup

Realiza el "setup" de openvas. Tras un proceso relativamente largo se abre de manera automática el explorador predeterminado con la web de Greenbone ya cargada esperando el registro. El usuario por defecto será "admin" mientras que la contraseña

se autogenerará al final del proceso de “setup”, en este caso la contraseña generada es “e01878ac-3871-41b2-9a28-092e339e0e96" que hay que copiar para poder acceder.

Ya estando dentro de la web, se selecciona “scans”, “tasks” y finalmente el botón de “wizard” el cuál tras introducir la dirección ip de la víctima realizará el análisis de forma automática, lo que genera el informe con el número de vulnerabilidades totales divididas en niveles de gravedad.

Una vez obtenido dicho informe es necesario acceder a Metasploit, para ello desde el terminal del Kali utilizaremos los siguientes comandos:

$ msfconsole
Permite el acceso al terminal del Metasploit.

Msf> load openvas
Carga los comandos necesarios para la obtención del reporte.

Msf> openvas_connect admin e01878ac-3871-41b2-9a28-092e339e0e96 localhost 9390
Conecta el Metasploit con el Greenbone al que accedimos previamente mediante un usuario (admin) y unaa contraseña (e01878ac-3871-41b2-9a28-092e339e0e96).

Msf> openvas_report_list
Extrae un listado con los reportes generados en dicho Greenbone, así como sus IDs, en este caso el ID generado es cae4bce4-4d4c-48c8-b575-cc9b4555d036.

Msf> openvas_format_list
Ofrece un listado de formatos compatibles como PDF, XML o TXT entre otros, cada uno con su respectivo ID, para este caso se eligio PDF, pero otras opciones son igualmente válidas. El ID de PDF es c402cc3e-b531-11e1-9163-406186ea4fc5.

Msf> openvas_report_import cae4bce4-4d4c-48c8-b575-cc9b4555d036 c402cc3e-b531-11e1-9163-406186ea4fc5
Se importa el reporte necesario a la base de datos de Metasploit, especificando el ID del informe y el ID del formato en el que se quiere importar.

Una vez que ya esté en la base de datos se puede utilizar la interfaz gráfica del Armitage. Una vez en la aplicación, hay que conectarse con las opciones que pone por defecto, y tras la conexión se busca las máquina víctima mediante “Host” / “MSF scans” para que la detecte.

Ya con la máquina detectada se hará un “Hail Mary”, es decir, un mapeo de todas las vulnerabilidades detectadas en la máquina, lo que permitirá lanzar todos los exploits posibles a dicho objetivo de forma sucesiva, dejando el equipo en una situación vulnerable. De esta forma la auditoria informática termina ofreciendo el listado de vulnerabilidades que dieron éxito y podrían ser de riesgo.

6.2 ANEXO B

En este caso, se explicará paso a paso cada fase, sin entrar en detalle al porqué que ya se detalló previamente en el punto 4.3 PENTESTING MEDIANTE CONTENEDORES.

En esta parte se intentará emular los mismos pasos que anteriormente, mediante la tecnología de contenedores, en sustitución de las máquinas. Para ello, se montará una máquina virtual, por seguridad, ya que no sería necesario, con el sistema operativo Ubuntu 18.04LTS por ser uno de los más actualizados de Linux, ya que esta tecnología es nativa de Linux y se hace más agradable su manejo. La descarga se realizará desde la página oficial de Ubuntu, lo que nos descargará un archivo .iso con todo lo necesario.
Para el montaje de la máquina habrá que seguir unos pasos parecidos a los de Metasploitable:
1. Seleccionar “Nuevo”.
2. Elegir nombre de la máquina (Ubuntu), ubicación de la máquina y sistema operativo (Linux) así como la versión (Ubuntu (64 bits)).
3. Asignar cuánta memoria RAM queremos destinar a la máquina (8Gb).
4. Crear disco duro virtual.
5. VDI (Virtual Disk Image).
6. Reservado dinámicamente.
7. Proporcionamos memoria a la máquina (100Gb).

Una vez con la máquina creada tendremos que acceder a la opción de almacenamiento de la máquina y asignaremos ese archivo .iso que acabamos de descargar. En este punto la máquina ya está preparada para su primer arranque y configuración. Una vez arrancada nos pide una configuración inicial con aspectos típicos como idioma, idioma de techado, zona horaria, usuario y contraseña y alguna actualización extra. Una vez terminado estará listo para usar como si fuese un ordenador independiente, por lo que iniciaremos la instalación de Docker y todos los aspectos necesarios.

Para instalar Docker, será necesario acceder a la web oficial [59] en la cual explican paso a paso como realizar la instalación específicamente para Ubuntu en nuestro caso. Hay que seguir dos fases para su instalación, la primera es hacer un set up del respositorio de Docker.Los pasos a seguir en el terminal de Ubuntu son:

```
$ sudo apt-get remove docker docker-engine docker.io containerd runc
```
Desinstalar las versiones antiguas de Docker que pudiese haber en el sistema operativo, para poder instalar la última.

```
$ sudo apt-get update
```
Actualizar el paquete apt.

```
$ sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```
Instalar los paquetes que permiten al apt usar repositorios sobre HTTPS:

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –
Añadir la clave oficial GPG de Docker.

$ sudo apt-key fingerprint 0EBFCD88
Antes de continuar se debe verificar que la clave con la huella es 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88

$ sudo add-apt-repository "deb [arch=amd64] \
https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
Finalmente se añade el repositorio.

La segunda fase consiste en la instalación del Docker CE. Los pasos a seguir son:

$ sudo apt-get update
Actualizar el paquete apt.

$ sudo apt-get install docker-ce docker-ce-cli containerd.io
Instalar la última versión de Docker CE

$ sudo docker run hello-world
Finalmente, como mera comprobación arrancar el contenedor ejemplo y verificar que funciona. Si todo salió correctamente saldrá algo similar a la figura 37.

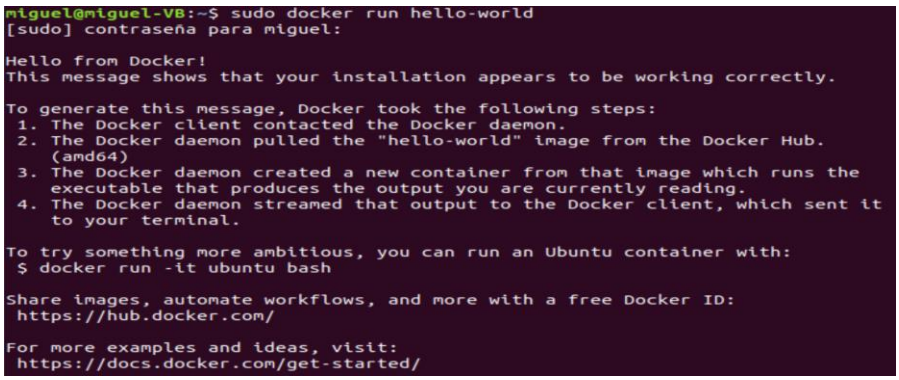

Figura 37: Pantalla de inicio del contenedor “Hello World”

Una vez instalado Docker lo siguiente será crear los contenedores adecuados, capaces de hacer todas las funcionalidades necesarias. En primer lugar se buscará un contenedor víctima basado en Metasploitable que ofrezca un número de vulnerabilidades considerable. El elegido en este caso se puede ver y descargar en [60]. Los pasos a seguir para su creación son:

$ sudo docker pull tleemcjr/metasploitable2
Con este comando se descargará la imagen sobre la que se basará el contenedor se desee crear.

$ sudo docker run –name meta2 – t – i tleemcjr/metasploitable2 /bin/bash

Este comando arrancará el contenedor y permitirá operar comandos dentro de él. Además al poner –name meta2 no es necesario memorizar el ID del contenedor, sino llamarle por el nombre asignado.

$ sudo docker ps –a

Listado de todos los contenedores con sus respectivos ID, para llevar un control sobre ellos y saber cuáles están arrancados.

En este punto ya se ha accedidio dentro del contenedor Metasploitable, y no es necesario hacer mucho más, aparte de descubrir su ip (172.17.0.3) con un "ifconfig", como muestra la figura 38, puesto que ya se buscó un contenedor que tuviese los servicios activos.

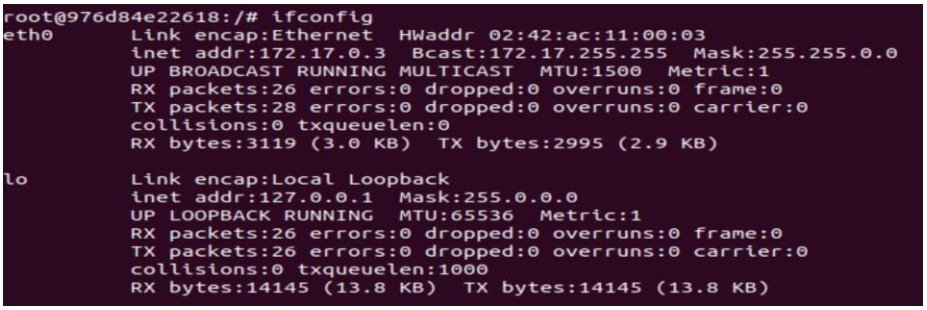

Figura 38: Contenedor arrancado viendo su IP

Por si se necesitase manipular este contenedor, o cualquier otro, es recomendable saberse estos comandos (ejecutarlos fuera del contenedor, en el terminal del Ubuntu):

$ sudo docker stop meta2

Permite parar el contenedor para que no esté funcionando continuamente.

$ sudo docker start –i meta2

Permite arrancar el contenedor ya creado previamente y poder manipularlo.

S sudo docker rm meta2

Permite borrar el contenedor si ya no lo necesitamos.

$ sudo docker exec –it meta2 /bin/bash

Permite acceder al terminal de un contenedor que ya está arrancado sin necesidad de pararlo.

A continuación se procederá con el contenedor atacante, el Kali, la parte más delicada del proyecto, puesto que es algo bastante novedoso, no hay mucho con lo que guiarse, por lo que ha habido mucha prueba y error. Finalmente se ha conseguido optimizar todo el proceso en un solo contenedor, lo que lo hace un proceso bastante eficiente. Los pasos a seguir se realizarán en varios terminales:

Terminal 1:

$ sudo docker pull simonthomas/armitage

Se eligió esta imagen porque resolvía uno de los problemas más significativos del proyecto, que era generar la interfaz gráfica de Armitage. Además está creado con base en Kali, por lo que en teoría es el mismo caso que con las máquinas virtuales.

$ id

Comprobar que el uid y el gid son 1000 para no tener problemas de accesos.

$ sudo xhost +

Desactiva el control de acceso, por lo que los equipos pueden conectarse desde cualquier host, cada vez que se arranque o cree este contenedor debe ejecutarse antes.

$ sudo docker run –name armitage –ti –e DISPLAY=$DISPLAY –v /tmp/.x11-unix:/tmp/.x11-unix simonthomas/armitage

Comando que arranca el contenedor y deja el terminal bloqueado para poder generar la ventana de Armitage. Conectarse al Armitage a través de la ventana generada, con los valores por defecto que aparecen, y no cerrar esta ventana en ninguna fase del proceso, ni el terminal. En este momento es necesario abrir otro terminal y acceder al contenedor que ya está funcionando. Esto sucede la primera vez al crearlo, cuando se arranca un contenedor de Armitage ya creado, no se bloquea el terminal y permite seguir trabajando en el mismo.

Terminal 2:

$ sudo docker exec –it armitage /bin/bash

Comando para acceder al terminal del contenedor que ya está arrancado.

$ apt-get update && apt-get dist-upgrade

Una vez dentro del terminal es recomendable actualizar como buena práctica.

$ apt-get update && apt-get install -y openvas

Instalación y actualización de los servicios de Openvas.

En este punto del proyecto hay que realizar una serie de comandos para corregir diferentes fallos de acceso que aparecen a lo largo del trabajo.

$ nano /etc/openvas/openvassd.conf

Modificar el kb_location a “var/run/redis-server.sock” como muestra la figura 39.

```
# Use location matching /etc/redis/redis-openvas.conf which is
# used by systemd's redis@openvas.service
kb_location = /var/run/redis-server.sock
```

Figura 39: Modificación del archivo openvassd.conf

$ nano /etc/redis/redis-openvas.conf

Modificar el contenido de unixsocket a “/var/run/redis-server.sock” y el contenido de unixsocketperm a “700” como muestra la figura 40.

```
# Unix socket.
#
# Specify the path for the Unix socket that will be used to listen for
# incoming connections. There is no default, so Redis will not listen
# on a unix socket when not specified.
#
unixsocket /var/run/redis-server.sock
unixsocketperm 700
```

Figura 40: Modificación del archivo redis-openvas.conf

$ openvas-setup

Realiza el “setup” de openvas. Tras un proceso relativamente largo, genera automáticamente la contraseña de openvas “2043201b-5613-4d0e-8e8c-61c83e8d4dfd", el usuario sigue siendo “admin”, abrirá la ventana de openvas pero no funcionará por una serie de errores que siguen apareciendo. Aun así es buen momento para hacer un guardado de todo lo que se ha avanzado, guardándolo en una imagen propia, puesto que de aquí en adelante, todos los pasos que se sigan habrá que repetirlos cada vez que se arranque el contenedor. Para guardar una imagen del momento exacto de nuestro contenedor, con todas las instalaciones y actualizaciones, se hará desde el terminal de Ubuntu con:

$ sudo docker commit armitage miguel/docker:version3

Este comando captura el momento del contenedor “armitage”, también se puede poner el ID del contenedor, y lo guarda en la imagen con nombre “miguel/docker:version3”.

En este punto se puede continuar con el contenedor ya creado o generar uno nuevo a partir de la imagen generada, es indiferente. Los pasos a seguir a partir de este punto para que funcione el Openvas, una vez dentro del contenedor, son:

$ redis-server /etc/redis/redis-openvas.conf

Activa el servidor redis en el archivo que modificamos prevfiamente

$ gsad -f--listen=127.0.0.1--mlisten=127.0.0.1--mport=9390 &

Comando propio de Openvas que permite indicar la dirección por la que escuchar, la dirección del gestor y el puerto.

$ netstat -antp

Permite comprobar que se ha generado el gsad y está escuchando en el puerto 443 y el 80, como muestra la figura 41.

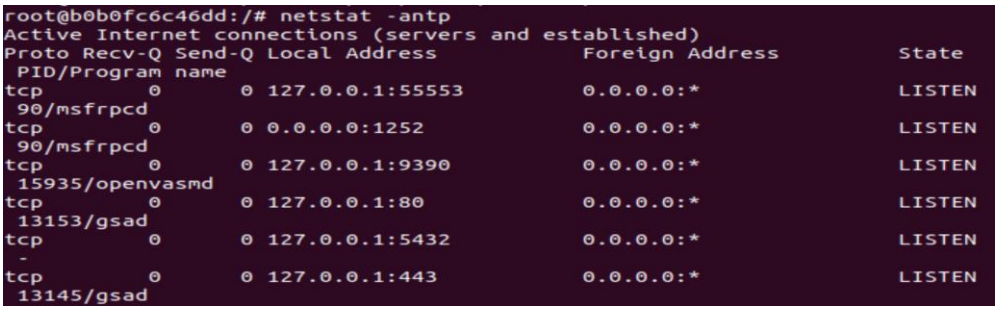

Figura 41: Resultados del netstat con el gsad en los puertps 443 y 80

$ openvas-start
Se arranca el proceso en el que se hicieron todas las instalaciones y descargas previamente, aun así dará un error o dos puesto que por defecto detiene el escáner de Openvas, e incluso puede que el manager, para saber si están iniciados están os siguientes comandos:

$ /etc/init.d/openvas-manager status
Comprueba si el gestor de Openvas está iniciado.

$ /etc/init.d/openvas-scanner status
Comprueba si el escáner de Openvas está iniciado.

Si cualquiera de los dos está parado habría que arrancarlo de la siguiente forma:

$ /etc/init.d/openvas-manager start
Arranca el gestor de Openvas
$ /etc/init.d/openvas-scanner start
Arranca el escáner de Openvas

En la ventana que generó el último “openvas-setup”, habrá que quitar de la URL el puerto por defecto que se pone y así se podrá acceder a la página oficial. Con el usuario y la contraseña anteriormente mencionados se puede acceder y realizar los mismos pasos que en las máquinas virtuales, atacando a la ip del contenedor Metasploitable. Ya en este punto, en el que se ha conseguido instalar todo, el proceso es prácticamente el mismo que en las máquinas virtuales.

Una vez dentro de la página de Greenbone seleccionar “scans”, “tasks” y finalmente el botón de “wizard”, se introduce la dirección ip de la víctima y tras unos cuantos minutos generará el reporte con un número similar de vulnerabilidades que en el caso anterior.

Una vez obtenido el informe, y antes de acceder a Metasploit, es recomendable realizar estos dos comandos:

$ /etc/init.d/postgresql start
Inicia el servicio sql

$ msfdb init
Inicia la base de datos de Metasploit

Una vez ya tomadas estas dos precauciones se accede a Metasploit con los comandos anteriormente mencionados y se importa el informe:

$ msfconsole
Permite el acceso al terminal del Metasploit.

Msf> load openvas
Carga los comandos necesarios para la obtención del reporte.

Msf> openvas_connect admin 2043201b-5613-4d0e-8e8c-61c83e8d4dfd localhost 9390
Conecta el Metasploit con el Greenbone al que se accedió previamente mediante el usuario y la contraseña.

Msf> openvas_report_list
Extrae un listado con los reportes generados en dicho Greenbone, así como sus IDs, en este caso el ID generado es b13f83ce-2d9b-4e9e-98b9-b97d0f77e6d5.

Msf> openvas_format_list
Ofrece un listado de formatos compatibles como PDF, XML o TXT entre otros, cada uno con su respectivo ID, para este caso se eligió PDF, pero otras opciones son igualmente válidas. El ID de PDF es c402cc3e-b531-11e1-9163-406186ea4fc5.

Msf> openvas_report_import b13f83ce-2d9b-4e9e-98b9-b97d0f77e6d5 c402cc3e-b531-11e1-9163-406186ea4fc5
Importar el reporte que se necesita a la base de datos de Metasploit, especificando el ID del informe generado y el ID del formato en el que se quiere importar.

Una vez que ya se tiene en la base de datos se puede utilizar la interfaz gráfica del Armitage, que se abrió desde que se arrancó el contenedor y no se debería de haber cerrado. Una vez en la aplicación, buscar el contenedor víctima mediante “Host” / “MSF scans” para que la detecte. Ya con el contenedor detectado hacer un “Hail Mary”, es decir, un mapeo de todas las vulnerabilidades detectadas en la víctima, lo que permitirá lanzar todos los exploits posibles a dicho objetivo, dejando el equipo en una situación vulnerable. Al igual que en las máquinas virtuales se podrán ejecutar exploits o acceder a sus shells.

En esta situación ya se habrá conseguido el objetivo que se buscaba, emular de la manera más exacta posible las posibilidades de Pentesting que se consiguieron en la parte de las máquinas virtuales.

6.3 ANEXO C

En este anexo, se explicará paso a paso cada fase, sin entrar en detalle al porqué, debido a que ya se detalló previamente en el punto 4.6 PRUEBA DE CONCEPTO DE PENTESTING PARA SU APLICACIÓN EN LOS LABORATORIOS DOCENTES.

En este anexo se hará una aproximación de los comandos necesarios para poder realizar las dos prácticas docentes anteriormente propuestas en el apartado 4.5. Esta idea es un breve planteamiento realizado desde el sistema operativo del equipo anfitrión hacia la máquina virtual que alberga los contenedores. Es un escenario a reducida escala, pero que permite ver el concepto de la idea que se plantea, desde el tipo de conexión que se utiliza, las herramientas empleadas o algunos de los comandos necesarios para su correcto funcionamiento.

Para la primera parte se recurrirá a la herramienta Putty para establecer la conexión SSH entre el Windows 10 del anfitrión y el Ubuntu de la máquina. Para descargar esta herramienta habrá que ir a la página [61], mostrando una interfaz como la mostrada en la figura 42.

Download PuTTY: latest release (0.72)

Home | FAQ | Feedback | Licence | Updates | Mirrors | Keys | Links | Team
Download: **Stable** · Snapshot | Docs | Changes | Wishlist

This page contains download links for the latest released version of PuTTY. Currently this is 0.72, released on 2019-07-20.

When new releases come out, this page will update to contain the latest, so this is a good page to bookmark or link to. Alternatively, here is a permanent link to the 0.72 release.

Release versions of PuTTY are versions we think are reasonably likely to work well. However, they are often not the most up-to-date version of the code available. If you have a problem with this release, then it might be worth trying out the development snapshots, to see if the problem has already been fixed in those versions.

Package files

You probably want one of these. They include versions of all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the FAQ entry.)

MSI ('Windows Installer')

32-bit:	putty-0.72-installer.msi	(or by FTP)	(signature)
64-bit:	putty-64bit-0.72-installer.msi	(or by FTP)	(signature)

Unix source archive

.tar.gz:	putty-0.72.tar.gz	(or by FTP)	(signature)

Figura 42: Página web de descarga de Putty

En ella habrá que seleccionar la opción de 64bits para el sistema operativo deseado, en este caso Windows. Una vez seleccionado, su instalación es mecánica, seleccionar todo por defecto y elegir el lugar del equipo donde se desee alojar el programa.

Antes de arrancarlo para establecer la conexión será necesario realizar unos pasos antes. En primer lugar dentro la máquina virtual, será necesario ejecutar un par de comandos desde la terminal:

$ sudo apt-get install openssh-server

Este comando instala un servidor SSH, así como todo lo necesario para establecer dicha conexión.

$ sudo /etc/init.d/ssh start
Arranca los servicios SSH para que la máquina sea capaz de recibir esa conexión.

A continución será necesario establecer la ruta entre el equipo anfitrión y la red donde se alojan los contenedores, para ello, desde la terminal del Windows en modo administrador será necesario crear esa ruta:

$ route add 172.17.0.0 mask 255.255.255.0 192.168.56.101
Este comando conecta el equipo con la red Docker (172.17.0.0) utilizando como gateway la red “Host Only” de Ubuntu (192.168.56.101).

Una vez realizados estos pasos se puede comprobar si verdaderamente es posible conectarse, para ello, una vez en la aplicación, y con la máquina encendida, será necesario especificar la ip a la que se quiere conectar, en este caso, la “Host-Only” de Ubuntu. Una vez dentro si la conexión se ha establecido correctamente aparecerá uma terminal solicitando el usuario y contraseña de la máquina virtual.

Ya en este punto se ha conseguido acceder a una terminal de la máquina, por lo que se puede realizar cualquier comando que se pueda ejecutar en la otra parte. Un ejemplo es el mostrado en la figura 43, donde a la izaquierda se muestra el resultado en el Putty, y a la derecha el resultado en la máquina, observando que son idénticos, como cabe esperar.

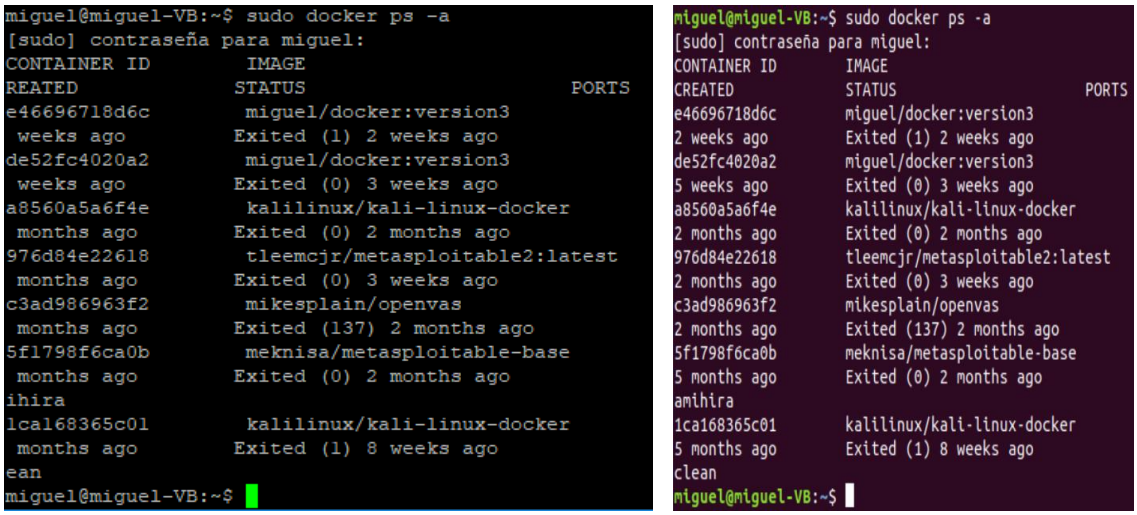

Figura 43: Ejemplo que prueba su correcto funcionamiento

Con esto ya es posible ejecutar todos los comandos del proyecto, pero hay partes como el análisis de vulnerabilidades de Openvas, que es necesario el navegador, para ello hay una serie de pasos a realizar que permiten conectarse a la dirección ip que deseamos.

Si se quiere conectar desde el explorador de la máquina virtual, es necesario modificar el archivo “redis-openvas.conf”, para ello habrá que utilizar, dentro del contenedor del kali, el comando:

$ sudo nano /etc/redis/redis-openvas.conf
Una vez en el archivo, comentar la línea del “bind”, tal y como muestra la figura 44.

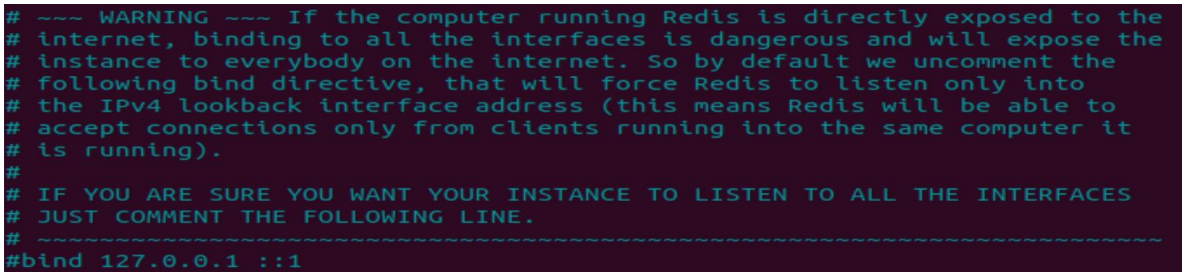

Figura 44: Modificación en redis-openvas.conf

Así mismo es necesario modificar el comando gsad utilizado anteriormente de la siguiente forma:

$ gsad -f --listen=172.17.0.2 --mlisten=127.0.0.1 --mport=9390 &
Esto permite que escuchen en la dirección ip del contenedor, lo que hace que se pueda conectar a esa dirección a través del navegador del Ubuntu.

Finalmente el último paso a seguir para que el navegador del equipo anfitrión sea capaz de conectar a la dirección del contenedor es un forwarding de puertos, para ello hay que realizar una serie de comandos en la terminal del Ubuntu:

$ sudo nano /etc/sysctl.conf
En este archivo hay que descomentar la línea que pone “net.ipv4.ip_forward=1”.

$ sysctl –system
Recarga el sysctl

$ sudo iptables list
Muestra la lista de iptables

$ sudo iptables –A FORWARD –i docker0 –j ACCEPT
$ sudo iptables –A FORWARD –i enp0s8 –j ACCEPT
Configura las iptables para realizar el forwarding entre la red de docker y el host only, permitiendo así acceder al openvas desde el explorador del Windows anfitrión

Finalmente con todo esto si se accede al navegador del equipo anfitrión y se introduce la dirección IP del contenedor (172.17.0.2), aparecerá la interfaz de Openvas, permitiendo así realizar el análisis de vulnerabilidades.

REFERENCIAS

[1] Novoselteva, E. (2018, 1 marzo). Top 10 Beneficios De Utilizar Docker. Recuperado de: https://apiumhub.com/es/tech-blog-barcelona/beneficios-de-utilizar-docker/

[2] Colaboradores de Wikipedia. (2019, 12 julio). Seguridad De La Información Aplicada A Computadoras Y Redes. Recuperado de: https://es.wikipedia.org/wiki/Seguridad_inform%C3%A1tica

[3] Ramiro, R. (2018, 3 enero). Algunos Tipos De Ataques Informáticos. Recuperado de: https://ciberseguridad.blog/algunos-tipos-de-ataques-informaticos/

[4] Prenafeta, J. (2018, 10 diciembre). Qué Es Pentesting Y Cómo Detectar Y Prevenir Ciberataque. Recuperado de: https://www.hiberus.com/crecemos-contigo/que-es-Pentesting-para-detectar-y-prevenir-ciberataques/

[5] Esaú, A. (2018, 24 octubre). Qué Es El Pentesting. Recuperado de: https://openwebinars.net/blog/que-es-el-Pentesting/

[6] Medrano, V. (2015, 23 agosto). Las Fases De Un Test De Penetración. Recuperado de: https://cyberseguridad.net/index.php/455-las-fases-de-un-test-de-penetracion-pentest-Pentesting-i

[7] Vulnerabilidad De Los Sistemas Informáticos. (2013, 11 abril). Recuperado de: http://vulnerabilidadtisg.blogspot.com/

[8] Gómez, H. (2019, 23 enero). Los Cuatro Pilares Clave De La Seguridad Cloud. Recuperado de: http://www.conocedell.es/cloud/los-cuatro-pilares-clave-de-la-seguridad-cloud

[9] Andrés, R. (2017, 31 mayo). Qué Es Una Máquina Virtual, Cómo Funciona Y Para Qué Sirve. Recuperado de: https://computerhoy.com/noticias/software/que-es-maquina-virtual-como-funciona-que-sirve-46606

[10] Ramírez, I. (2016, 5 agosto). Máquinas Virtuales: Qué Son, Cómo Funcionan Y Cómo Utilizarlas. Recuperado de: https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-funcionan-y-como-utilizarlas

[11] Colaboradores de Arsys. (2019, 14 marzo). ¿Qué Es Docker Y Qué Ventajas Tiene Trabajar Con Sus Contenedores? Recuperado de: https://www.arsys.es/blog/soluciones/docker-ventajas-contenedores/

[12] Alarcón, J. M. (2018, 14 junio). ¿Qué Diferencia Hay Entre Docker Y Máquinas Virtuales? Recuperado de: https://www.campusmvp.es/recursos/post/que-diferencia-hay-entre-docker-contenedores-y-maquinas-virtuales.aspx

[13] Tanjim, M. (2019, 11 mayo). Best Linux Distributions For Hacking And Penetration Testing. Recuperado de: https://itsfoss.com/linux-hacking-penetration-testing/

[14] Our Most Advanced Penetration Testing Distribution, Ever.. (s.f.). Recuperado de https://www.kali.org/

[15] Faletra, L. (s.f.). Discover the Parrot Universe and get the most from our awesome Debian-based platform.. Recuperado de https://parrotlinux.org/

[16] BACKBOX.ORG. (s.f.). Recuperado de https://www.backbox.org/

[17] Samurai Web Testing Framework. (s.f.). Recuperado de http://www.samurai-wtf.org/

[18] Pentoo. (s.f.). Recuperado de https://www.pentoo.ch/

[19] DEFT Linux. (s.f.). Recuperado de www.deftlinux.net

[20] CAINE Computer Forensics Linux Live Distro. (s.f.). Recuperado de https://www.caine-live.net/

[21] Network Security Toolkit (NST 30). (s.f.). Recuperado de https://www.networksecuritytoolkit.org/nst/index.html

[22] BlackArch Linux - Penetration Testing Distribution. (s.f.). Recuperado de https://blackarch.org/

[23] Bugtraq Team. (s.f.). Recuperado de http://bugtraq-team.com/

[24] Recopilación De Máquinas Virtuales Para Pentesting. (2014, 10 febrero). Recuperado de: https://www.adslzone.net/foro/seguridad-informatica.111/recopilacion-maquinas-virtuales-Pentesting.359536/

[25] PentesterLab. (s.f.). Recuperado de https://pentesterlab.com/

[26] Metasploit Unleashed Requirements. (s.f.). Recuperado de https://www.offensive-security.com/metasploit-unleashed/requirements/

[27] Hackxor. (s.f.). Recuperado de https://hackxor.net/

[28] Kioptrix ~ VulnHub. (s.f.). Recuperado de https://www.vulnhub.com/series/kioptrix,8/

[29] NETinVM. (s.f.). Recuperado de www.netinvm.org

[30] UltimateLAMP. (s.f.). Recuperado de https://www.vulnhub.com/entry/ultimatelamp-02,36/

[31] Colaboradores de SIAG Consulting. (2018, 10 septiembre). Los 5 Software De Virtualización Más Utilizados. Recuperado de: https://siagconsulting.es/5-software-virtualizacion/

[32] VMware España: cloud, movilidad, red y seguridad. (s.f.). Recuperado de https://www.vmware.com/es.html

[33] Citrix. (s.f.). Recuperado de https://www.citrix.es/

[34] Cree máquinas virtuales Linux o Windows con su cuenta gratuita de A.... (s.f.). Recuperado de https://azure.microsoft.com/es-es/free/virtual-machines/search/?

[35] Oracle VM VirtualBox. (s.f.). Recuperado de https://www.virtualbox.org/

[36] KVM. (s.f.). Recuperado de https://www.linux-kvm.org/page/Main_Page

[37] Doerrfeld, B. (2019, 21 enero). 5 Container Alternatives To Docker. Recuperado de: https://containerjournal.com/2019/01/22/5-container-alternatives-to-docker/

[38] CoreOS. (s.f.). Recuperado de https://coreos.com/rkt/docs/latest/

[39] Apache Mesos. (s.f.). Recuperado de http://mesos.apache.org/documentation/latest/containerizers/

[40] Linux Containers. (s.f.). Recuperado de https://linuxcontainers.org/

[41] Open source container-based virtualization for Linux.. (s.f.). Recuperado de https://openvz.org/

[42] Containerd. (s.f.). Recuperado de https://containerd.io/

[43] Colaboradores de campusMVP. (2018, 7 agosto). Las 10 Herramientas Más Importantes Para Orquestación De Contenedores Docker. Recuperado de: https://www.campusmvp.es/recursos/post/las-10-herramientas-mas-importantes-para-orquestacion-de-contenedores-docker.aspx

[44] Orquestación de contenedores para producción. (s.f.). Recuperado de https://kubernetes.io/es/

[45] Swarm mode overview. (s.f.). Recuperado de https://docs.docker.com/engine/swarm/

[46] Nubersia. (s.f.). Recuperado de https://www.nubersia.com/es/blog/mesosphere-dcos/

[47] Azure Containers | Microsoft Azure. (s.f.). Recuperado de https://azure.microsoft.com/es-es/product-categories/containers/

[48] AWS | Gestión de contenedores (ECS) compatible con los de Docker. (s.f.). Recuperado de https://aws.amazon.com/es/ecs/

[49] García, A. E. Práctica De Laboratorio: Pentesting Of Cloud Services

[50] Duarte, E. (2016, 13 abril). Las 8 Mejores Herramientas De Seguridad Y Hacking. Recuperado de: http://blog.capacityacademy.com/2012/07/11/las-8-mejores-herramientas-de-seguridad-y-hacking/

[51] Colaboradores de Greenbone. (s.f.). Openvas - Open Vulnerability Assessment Scanner. Recuperado de: http://www.openvas.org/

[52] DragonJAR, E. (2014, 15 noviembre). Manual De Armitage. Recuperado de: https://www.dragonjar.org/manual-de-armitage-en-espanol.xhtml

[53] Colaboradores de Wikipedia. (2019, 9 julio). Redis. Recuperado 14 julio, 2019, de: https://es.wikipedia.org/wiki/Redis

[54] Colaboradores de Wikipedia. (2019c, 4 julio). Secure Shell. Recuperado de: https://es.wikipedia.org/wiki/Secure_Shell

[55] Colaboradores de Kali Linux. (s.f.). Kali Linux Downloads. Recuperado de https://www.kali.org/downloads/

[56] Kali Linux Downloads – Virtual Images. (s.f.). Recuperado de https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/

[57] Ahmadi, M. (2019, 10 julio). How to Download and Install Metasploitable in VirtualBox. Recuperado de https://www.wikigain.com/download-install-metasploitable-in-virtualbox/

[58] Metasploitable Virtual Machine to Test Metasploit. (s.f.). Recuperado de https://information.rapid7.com/download-metasploitable-2017.html

[59] Colaboradores de Docker. (2019, 12 agosto). Get Docker Engine - Community for Ubuntu. Recuperado de https://docs.docker.com/install/linux/docker-ce/ubuntu/

[60] Docker Hub. (s.f.). Recuperado de https://hub.docker.com/r/tleemcjr/metasploitable2

[61] Download PuTTY: latest release (0.72). (s.f.). Recuperado de https://www.chiark.greenend.org.uk/%7Esgtatham/putty/latest.html